ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ т. 12, № 2(49), с. 53–71

ББК 32.811.1:22.192.23
ГРНТИ 50.07.03
УДК 004.942+004.272.3

Е. А. Аксёнова, А. А. Лазутина, А. В. Соколов

Минимизация средних затрат на перераспределение при работе с work-stealing деком в двухуровневой памяти

Аннотация. В работе рассмотрена задача оптимального управления work-stealing деком (англ. — deque) в двухуровневой памяти. Предполагается, что известны вероятности параллельных операций с деком и временные характеристики памяти для двух уровней. Задача состоит в нахождении оптимального числа элементов с двух сторон дека, которые при перераспределении дека должны быть оставлены в быстрой памяти. В качестве критерия оптимальности рассмотрены минимальные средние затраты на перераспределение памяти, которые возникают в случае переполнения или опустошения быстрой памяти. Такой критерий позволяет учитывать конкретные скорости доступа к уровням памяти и применять разработанные методы к разным сочетаниям быстрой и медленной памяти. Построены математическая и имитационная модели процесса работы с деком, представлены результаты численных экспериментов.

Ключевые слова и фразы: work-stealing балансировщики, work-stealing деки, кэширование деков, случайные блуждания, имитационные модели.

Введение

Хорошо известны аппаратные и программные механизмы кэш-памяти и виртуальной памяти, когда обмены между уровнями памяти при переполнении быстрой памяти происходят автоматически. Кэш и виртуальная память пытаются работать одинаково со всеми структурами данных и программами. Понятно, что универсальный механизм не может быть использован во всех ситуациях. Например, в операционных системах реального времени механизм виртуальной памяти не используется, так как не может обеспечить ограничения по времени. Для

Выполнена при финансовой поддержке РФФИ, проект №18-01-00125-а.

10.25209/2079-3316-2021-12-2-53-71

некоторых наиболее важных структур данных известны альтернативные реализации.

1. Обзор методов реализации структур данных

При переполнении или опустошении регистровой вершины стека в стековых компьютерах применяются разные программные и аппаратные методы [1–4]. Приведем некоторые методы реализации стековых архитектур.

(1) **Large Stack.** Можно сделать стек большим и предполагать, что переполнений не будет. Если переполнение произойдет, то необходимо совершить аварийное завершение работы. Такая реализация используется, например, в процессорах WISC CPU/16 (Wisconsin Integrally Synchronized Computer Central Processing Unit) с размером стека 256 элементов. Так как одно из основных приложений стековых компьютеров – управление в реальном времени, то такое решение чаще всего неприемлемо.

(2) **Demand Fed Single Element Stack Manager.** В данной стратегии вершина стека реализована аппаратно как циклический буфер, продолжение стека находится в памяти. На каждый стек требуется три указателя: на верх и низ аппаратного стека, и на верх продолжения стека в памяти. При переполнении и опустошении аппаратного стека всегда перемещается один элемент в память из буфера или наоборот. Метод реализован в FRISK 3 [3].

(3) **Paging Stack Manager.** Здесь вершина реализована программно как часть специальной памяти. При опустошении вершины стека из памяти копируется половина буфера, при переполнении вершины нижняя половина буфера перемещается в память, верхняя половина – в начало буфера. Метод реализован в RTX 2000 и RTX 32P (RTX – Real Time eXpress) [3].

(4) **An Associative Cache.** Кэш-память для стековых машин не дает преимуществ по сравнению с рассмотренными методами, так как сложнее для аппаратной реализации и не учитывает специфики стека как структуры LIFO. Если в стек нужно поместить структуры данных переменного размера, такие как строки и структуры, использование кэш-памяти может быть целесообразным [3].

(5) **Barometer-Pointer Algorithm.** Предлагаются реализации стека, когда перемещения элементов между уровнями памяти происходят не при переполнениях или опустошениях, а при

отклонении от оптимального состояния вершины в фоновом режиме (считается, что это половина быстрого буфера), т.е. параллельно с обычным выполнением команд [2].

(6) **Продолжение стека.** В работе [1] предлагается реализация стека, когда при переполнении и опустошении быстрой вершины переносится k элементов стека. В данных методах вершина стека является продолжением стека, находящегося в памяти второго уровня.

(7) **Копия стека.** Возможны аппаратные методы реализации, в которых вершина стека является копией стека, находящегося в памяти второго уровня. Тогда запоминания элементов при переполнении не требуется, достаточно лишь восстановления при опустошении.

(8) В системе Эльбрус [4] каждый процесс может иметь стек процедур (Procedure Stack – PS), стек связующей информации (Procedure Chain Stack – PCS) и стек пользователя (User Stack – US). В случае работы с несколькими параллельными процессами, каждый из которых имеет три стека, возникает проблема размещения стеков в общей памяти. Стеки пользователя для всех процессов размещаются в памяти ядра операционной системы. Рассмотрим задачи, которые возникают при различных способах размещения этих трех стеков.

- Если два стека PS и PCS располагаются в пространстве пользователя, то возникают задачи оптимальной работы со стеками пользователя разных процессов в общей памяти ядра, со стеками процедур и стеками связующей информации разных процессов в общей памяти пользователя.
- Если все три стека располагаются в пространстве ядра операционной системы, то для каждого процесса возникает задача размещения трех стеков в общей памяти – какие стеки расположить навстречу друг другу, а какой отдельно; как разделить память между стеками, растущими навстречу друг другу, и третьим стеком.
- Если какие-то из этих стеков имеют регистровую вершину, то возникают задачи организации работы с этими стеками в двухуровневой памяти, т. е. перераспределения стеков после переполнения и опустошения быстрой памяти. Эту задачу нужно решать для одного стека, для двух стеков, растущих навстречу друг другу в общей быстрой памяти, и для трех стеков, расположенных в общей быстрой памяти.
- Также в системе Эльбрус, как и некоторых других архитектурах, например, в AMD29K использовался способ

работы с объектами разных длин, когда они записываются не в «кучу» (heap), а в стеки. Для таких стеков нужно рассматривать модели, в которых элементы стека могут быть переменной длины.

В настоящее время появляются новые важные приложения, в которых используются стековые виртуальные машины. Например, в работе [5] дано описание виртуальной машины Etherbox32vm, предназначенной для реализации сценариев функционирования узлов сенсорной сети.

Для динамической балансировки параллельных вычислений в общей памяти часто используют стратегию «work-stealing», при которой пустые ядра забирают задачи у других [6]. У каждого ядра есть дек (англ. – deque), в котором располагаются указатели на задачи ядра или объекты данных задач. Если возникает новая задача, ядро добавляет указатель (объект) этой задачи в свой дек; если ядру нужна задача, оно читает указатель (объект) своего дека. Если ядро определяет, что его дек пустой, оно перехватывает («крадет») указатели (объекты) у другого ядра. Операции включения и исключения объектов ядром выполняются на одном конце, а перехват («кража») объектов другими ядрами происходит на другом конце дека. Такая структура данных называется «дек с ограниченным входом» [7]. Например, можно перехватывать один элемент или половину. Стратегия «work-stealing» используется в таких языках программирования, как Cilk, Cilk++, библиотеке параллельных задач TPL (Task Parallel Library), кроссплатформенной библиотеке шаблонов TBB (Threading Building Blocks), X10, JSR 166 (Java Specification Requests) (пакет java.util.concurrent), Erlang, стандартах OpenMP (Open Multi-Processing) в реализации компиляторов ICC (Intel C++ Compiler).

В работах [8, 9] предложены модели для оптимального управления деками в памяти одного уровня для последовательного циклического представления деков. Разработан новый метод представления деков в памяти одного уровня, когда они двигаются друг за другом в одном участке общей памяти [10]. В статьях [11, 12] данный метод проанализирован для работы с двумя и тремя деками. Описание такого метода для FIFO-очередей представлено в работе [13], а в [14] проведен его анализ. Как показал опыт программной реализации, большое значение имеет оптимизация работы с кэш-памятью в параллельных балансировщиках задач, работающих по стратегии «work-stealing». Например, в реализации [15] за счет того, что в деках хранились объекты задач вместо указателей на задачи, удалось уменьшить

время выполнения тестовых задач в 2.5 раза и кэш-промахи до 30% по сравнению с work-stealing балансировщиками Intel TBB и Intel/MIT Cilk (MIT – Massachusetts Institute of Technology). Поэтому важно исследовать специальные методы работы с деками в двухуровневой памяти, а не только пытаться эффективно работать с универсальными реализациями кэш-памяти.

Нужно отметить, что если для стеков было предложено много различных методов программной и аппаратной реализации, то методы аппаратной реализации для деков авторам статьи неизвестны. Это можно объяснить тем, что деки начали использоваться на практике гораздо позже стеков, так как они стали нужны только в связи с появлением параллельного программирования. На практике для деков можно применять аналоги некоторых методов, которые применялись в компьютерах со стековой архитектурой. Например, можно применять модификацию метода «Barometer-Pointer Algorithm», в котором перемещения элементов между уровнями памяти происходят не при переполнениях или опустошениях, а при отклонении от половины быстрого буфера в фоновом режиме, т.е. параллельно с обычным выполнением команд. Только перемещения нужно начинать производить при отклонении от оптимального состояния, которое мы ищем.

В теории виртуальной памяти и кэш-памяти доказано, что при переполнении быстрой памяти оптимально перемещать в медленную память те данные, которые как можно дольше не понадобятся, но на практике это редко когда применяется. Для наиболее часто используемых структур данных можно предложить свои, отличные от универсальных, методы работы в двухуровневой памяти, например, регистры – оперативная память. В случае переполнения стека в быстрой памяти необходимо элементы из основания стека переносить в медленную память, так как они не понадобятся в ближайшем будущем (работа будет идти с вершиной стека). В случае переполнения очереди в медленную память следует переносить элементы из более новой части очереди, так как они пришли позже и доступ к ним тоже произойдет позже.

В статье [**16**] была предложена математическая модель для оптимального управления FIFO-очередью, а в [**17**, **18**] для оптимального управления одним и двумя стеками, растущими навстречу друг другу, в двухуровневой памяти. В [**19**] рассмотрен вариант задачи оптимального управления деком в двухуровневой памяти, когда в качестве критерия оптимальности используется критерий, который минимизирует среднее число перераспределений памяти. Эти задачи

являются продолжением исследований, начало которым было положено попытками решения задачи Д. Кнута [7].

Пусть в памяти объема m расположены два стека, растущие навстречу друг другу. В этом случае место их встречи можно рассматривать как случайную величину. Пусть известно, что на каждом шаге с вероятностью p может произойти включение информации в один из стеков и с вероятностью $(1-p)$ может произойти исключение элемента из одного из стеков. Обозначим через $M(m,p)$ математическое ожидание случайной величины $max(k_1,k_2)$, где k_1 и k_2 – длины стеков при встрече. Для безотказной работы в случае раздельного хранения стеков потребовалось бы $2max(k_1,k_2)$ единиц памяти, в то время как для стеков, направленных навстречу друг другу, нужно $m=k_1+k_2$ единиц памяти. В связи с этим Д. Кнут [7, 2.2.2, упр. 13] поставил задачу разработать математическую модель этого процесса и установить вид функции $M(m,p)$.

В работах [20–25] были построены математические модели для двух стеков, растущих навстречу друг другу, в виде двумерного случайного блуждания в треугольной области с двумя отражающими экранами и одним поглощающим экраном, разными методами решалась задача Д.Кнута.

В данной работе рассмотрены математическая и имитационная модели для оптимального управления work-stealing деком в двухуровневой памяти.

2. Модель для работы с деком в двухуровневой памяти

Рассмотрим последовательный циклический метод представления дека в быстрой памяти размера m. Операции включения и исключения элементов выполняются на одном конце дека, а «кражи» (операции исключения) – на другом конце. Принцип управления деком состоит в том, чтобы в быстрой памяти хранить элементы из концов дека, т. к. с ними ведется работа, а в медленной памяти хранить среднюю часть дека. При таком способе работы дек может иметь размер больше, чем размер быстрой памяти m.

При переполнении быстрой памяти часть элементов из середины дека перемещается в медленную память, а в быстрой памяти остаются элементы из концов дека. При опустошении одного из концов дека в быстрой памяти часть элементов перемещается из медленной в быструю память, если медленная память не пуста.

Обозначим x конец дека, на котором происходят включения элементов, а y конец дека, на котором происходят «кражи» элементов. После перераспределения получаем новый дек в быстрой памяти (рисунок 1), в котором x_0 элементов из конца, где происходят включения элементов, и y_0 элементов из конца, где происходят «кражи».

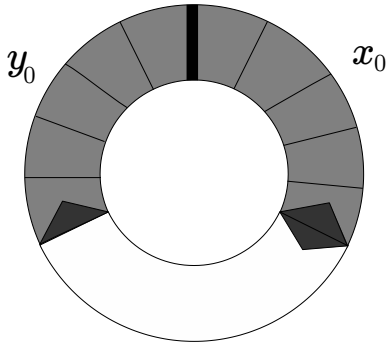

Рисунок 1. Дек в быстрой памяти после перераспределения

В случае переполнения дека или опустошения одного из концов происходит перераспределение элементов так, чтобы в быстрой памяти всегда оставались x_0 и y_0 элементов из концов дека. Нужно определить такие значения x_0 и y_0 элементов концов дека, которые следует оставлять в быстрой памяти при перераспределении, чтобы средние затраты на перераспределение памяти были минимальными.

Предположим, что операции с деком выполняются параллельно. Пусть p_1 и q_1 – вероятности включения и исключения элементов в дек на одном конце; q_2 – вероятность «кражи» элемента дека (исключения на другом конце); r – вероятность того, что длина дека не изменилась (возможно выполнение подзадачи или операции с другими деками); q_{12} – вероятность одновременного исключения с обоих концов дека (удаление элемента и одновременно «кража» элемента другим деком); pq_{12} – вероятность одновременного включения элемента и «кражи» элемента другим деком.

Процесс работы с деком после перераспределения можно описать в виде случайного блуждания по целым точкам (x, y) внутри треугольной области $x + y \leq m + 1$, $x \geq -1$, $y \geq -1$ (x – число элементов из конца дека, где происходят включения элементов; y – число элементов из конца дека, где происходят «кражи» элементов).

Точки на прямых $x+y = m+1$, $x = -1$ и $y = -1$ – это поглощающие состояния случайного процесса. Попадание случайного процесса

в поглощающее состояние означает перераспределение памяти, т. к. попадание на прямую $x + y = m + 1$ соответствует переполнению дека, а попадание на прямые $x = -1$ или $y = -1$ соответствует опустошению одного из концов дека.

Переходы в случайном блуждании будут происходить по схеме:

- из $(x, y) \to (x + 1, y)$ с вероятностью p_1,
- из $(x, y) \to (x - 1, y)$ с вероятностью q_1,
- из $(x, y) \to (x, y)$ с вероятностью r,
- из $(x, y) \to (x, y - 1)$ с вероятностью q_2,
- из $(x, y) \to (x - 1, y - 1)$ с вероятностью q_{12},
- из $(x, y) \to (x + 1, y - 1)$ с вероятностью pq_{12}.

Для каждой начальной точки (x, y) из области $x + y \leq m$, $x \geq 0$, $y \geq 0$ будем моделировать случайное блуждание и вычислять оценки вероятностей поглощения процесса на прямых $x + y = m + 1$, $x = -1$ и $y = -1$ с помощью имитационной модели. Вычисленные оценки соответствуют вероятностям переполнения и опустошения дека, если в быстрой памяти оставались x и y элементов из концов дека. Другими словами, это вероятности перераспределения памяти. Данные вероятности требуются для вычисления критерия оптимальности.

В задаче управления одной последовательной FIFO-очередью в двухуровневой памяти [16] рассмотрен частный случай такой модели для критерия оптимальности, который минимизирует среднее число перераспределений памяти (максимизирует среднее время до перераспределения памяти при переполнении или опустошении структуры данных). В этой задаче все вероятности, кроме p_1 и q_2, равны нулю, $x = 0$. Задача решена с использованием аппарата разностных уравнений [26], которые можно применить для среднего времени блуждания и решить рекуррентно, так как граничные условия позволяют это сделать. В случае управления деками такой способ построения модели применить нельзя. Заметим, что в случае, если нулю равны все вероятности, кроме p_1 и q_1, получаем задачу управления одним стеком в двухуровневой памяти [17], где $y = 0$.

3. Критерий оптимальности

Для вычисления предложенного критерия оптимальности необходимо знать вероятности перераспределения памяти и значения затрат времени на перераспределение. Для решения задачи построим имитационную модель, с помощью которой будем моделировать процесс случайного блуждания.

Для каждой точки (x, y) из области блуждания $x + y \leq m$, $x \geq 0$, $y \geq 0$ будем считать количество попаданий случайного процесса в каждое поглощающее состояние (количество перераспределений памяти). Если разделить полученные значения на число экспериментов N, то получим оценки вероятностей поглощения (перераспределения памяти) во всех поглощающих состояниях для каждой точки области $x + y \leq m$, $x \geq 0$, $y \geq 0$.

Рассмотрим стоимость перераспределения. Пусть t_1 – время перемещения одного элемента в быстрой памяти (например, регистры). Будем считать, что $t_1 = 1$ (такт), $t_2 = kt_1$ время обмена одним элементом между уровнями памяти, где k в настоящее время принимает значения от 100 до 1000. Будем считать, что пересылка z элементов в быстрой памяти занимает zt_1 единиц времени, а пересылка z элементов из быстрой памяти в медленную память или из медленной памяти в быструю память занимает $zt_2 = zkt_1$ единиц времени.

Рассмотрим состояние переполнения (x_1, y_1) на рисунке 2, в котором $x_1 > x_0$ и $0 < y_1 < y_0$. Пусть $c_{(x_1,y_1)}$ – это стоимость перераспределения в состоянии переполнения (x_1, y_1). Для перераспределения памяти и перехода к начальному состоянию (x_0, y_0) нужно переместить $(x_1 - x_0)$ элементов дека в медленную память и сдвинуть x_0 элементов в быстрой памяти в место соединения концов дека; переместить $(y_0 - y_1)$ элементов из медленной в быструю память в место соединения концов дека, сдвинув при этом y_1 элементов в быстрой памяти (рисунок 2). Тогда стоимость перераспределения $c_{(x_1,y_1)}$ в состоянии (x_1, y_1) для начального состояния (x_0, y_0) вычисляется так: $c_{(x_1,y_1)} = (x_1 - x_0)kt_1 + x_0t_1 + (y_0 - y_1)kt_1 + y_1t_1$. Применив аналогичные рассуждения, можно вычислить стоимость перераспределения в случае опустошения одного из концов дека.

Для вычисления критерия оптимальности определим средние затраты времени C на перераспределение памяти. Для каждого начального состояния (x, y), в котором $x + y \leq m$, $x \geq 0$, $y \geq 0$, вычислим средние затраты времени на перераспределение памяти по формуле $C = \sum_{i=1}^{n} p_i c_i$, где p_i – вероятность перераспределения в i-м состоянии перераспределения (вероятность попадания в i-е поглощающее состояние), c_i – стоимость перераспределения в i-м состоянии перераспределения, n – количество состояний перераспределения (поглощающих состояний). Для вычисления средних затрат будем использовать вычисленные с помощью имитационной программы оценки вероятностей перераспределения. Для решения задачи выберем такую точку (x_0, y_0), в которой средние затраты времени C на перераспределение памяти минимальны.

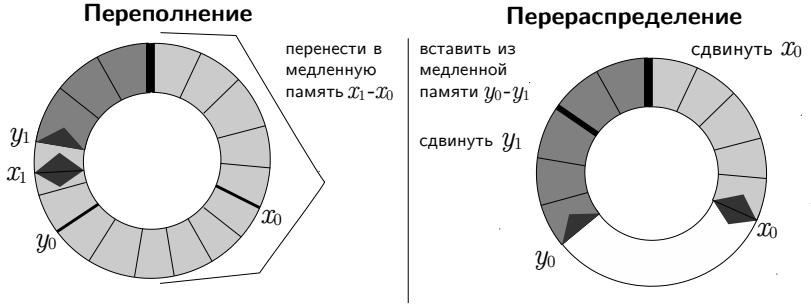

Рисунок 2. Перераспределение памяти в случае переполнения дека

В случае управления несколькими деками в двухуровневой памяти в качестве критерия оптимальности можно использовать некий суммарный критерий, например, минимальную сумму средних затрат для всех деков.

4. Некоторые численные результаты

В таблицах приведены численные результаты для двух критериев оптимальности: $Min\ C$ и $Min\ C/T$, где C – это средние затраты на перераспределение дека при переполнении или опустошении быстрой памяти, а T – это среднее время работы с деком до перераспределения памяти. Численные эксперименты проводились на процессоре Intel CORE i5, ОС Linux, компилятор gcc. Для получения результатов производилось по одному запуску программы. Величина C измерялась в тактах, величина T измерялась в шагах блуждания.

Критерий $Min\ C$ минимизирует средние затраты на перераспределение при переполнении или опустошении быстрой памяти, но при этом не учитывает время работы с деком до перераспределения памяти. Логично, что время работы с деком до перераспределения нужно максимизировать, т.к. в случае частого перераспределения памяти суммарные затраты на перераспределения будут возрастать.

Критерий $Min\ C/T$ минимизирует средние затраты на перераспределение дека при переполнении или опустошении быстрой памяти и в то же время максимизирует среднее время работы с деком до перераспределения памяти, то есть минимизирует среднее число перераспределений дека. Поясним, что $1/T$ это введенный в теории кэш-памяти средний коэффициент потерь [27], который равен процентному

отношению обращений к медленной памяти к общему числу обращений к памяти. Например, если в среднем 1% обращений производится к медленной памяти, а 99% – к быстрой, то значение $1/T = 0.01$. Результаты экспериментов с моделями подтверждают эти выводы.

Результаты вычислений показывают, что среднее время работы с деком до перераспределения для критерия $Min\ C/T$ может быть в десятки раз больше, чем среднее время работы с деком до перераспределения для критерия $Min\ C$. Следовательно, при использовании критерия $Min\ C$ перераспределение памяти будет происходить часто.

Таблица 1. $p_1 = 0.5$, $q_1 = 0.1$, $q_2 = 0.1$, $q_12 = 0.1$, $pq_{12} = 0.1$, $r = 0.1$, $k = 100$, $t_1 = 1$

m	$MinC$	T	x	y	$MinC/T$	T	x	y
5	208.52	1.16	3	0	49.67	5.4	1	2
10	110.66	1.15	10	0	27.14	13.09	3	4
15	116.73	2.29	14	0	22.38	19.01	4	5
20	118.55	1.15	19	0	17.98	27.99	4	9
25	145.85	1.92	21	0	15.85	32.49	4	10
30	148.95	1.16	2	0	13.85	41.04	4	14
35	144.31	2.33	32	0	2.88	49.94	6	16
40	115.74	1.92	40	0	11.92	58.5	6	19

Таблица 2. $p_1 = 1/6$, $q_1 = 1/6$, $q_2 = 1/6$, $q_12 = 1/6$, $pq_{12} = 1/6$, $r = 1/6$, $k = 100$, $t_1 = 1$

m	$MinC$	T	x	y	$MinC/T$	T	x	y
5	91.54	0.91	1	0	41.18	4.59	3	2
10	87.74	1.0	9	0	26.43	10.54	3	4
15	88.83	1.0	4	0	19.12	14.65	6	7
20	88.6	1.0	4	0	15.38	22.28	7	10
25	88.74	1.0	2	0	13.55	26.8	9	14
30	88.8	1.0	30	0	11.96	34.71	9	17
35	88.54	1.0	3	0	10.93	38.68	10	22
40	88.52	1.0	3	0	9.81	48.52	13	24

Заключение

В статье рассматривалась задача управления work-stealing деком в двухуровневой памяти. Для описания процесса работы с деком была построена имитационная модель в виде случайного блуждания по целочисленной решетке внутри треугольной области. В качестве

Таблица 3. $p_1 = 0.1$, $q_1 = 0.1$, $q_2 = 0.5$, $q_12 = 0.1$, $pq_{12} = 0.1$, $r = 0.1$, $k = 100$, $t_1 = 1$

m	$MinC$	T	x	y	$MinC/T$	T	x	y
5	101.55	0.43	5	0	34.19	5.22	2	3
10	75.84	0.41	5	1	20.07	8.77	3	5
15	46.15	0.41	2	0	15.21	13.12	4	9
20	46.13	0.43	2	0	13.05	16.13	6	13
25	46.17	0.43	3	0	11.19	21.8	6	16
30	49.44	0.43	19	0	9.74	30.26	8	21
35	46.13	0.43	3	0	8.94	35.97	9	25
40	49.22	0.43	18	0	8.34	42.98	10	27

Таблица 4. $p_1 = 0.1$, $q_1 = 0.1$, $q_2 = 0.1$, $q_12 = 0.5$, $pq_12 = 0.1$, $r = 0.1$, $k = 100$, $t_1 = 1$

m	$MinC$	T	x	y	$MinC/T$	T	x	y
5	56.78	0.43	5	0	50.11	3.17	3	2
10	59.68	0.43	9	0	27.83	7.23	4	5
15	59.88	0.41	4	0	21.54	10.19	6	7
20	57.67	0.43	20	0	17.52	14.41	10	9
25	59.75	0.43	5	0	15.69	18.58	11	11
30	59.74	0.43	2	0	13.77	21.68	13	15
35	60.03	0.43	8	0	12.84	25.8	15	17
40	59.75	0.43	4	0	12.02	26.12	17	22

Таблица 5. $p1 = 0.1$, $q_1 = 0.5$, $q_2 = 0.1$, $q_12 = 0.1$, $pq_12 = 0.1$, $r = 0.1$, $k = 100$, $t_1 = 1$

m	$MinC$	T	x	y	$MinC/T$	T	x	y
5	97.24	1.34	0	5	42.01	3.42	2	2
10	95.33	1.46	0	8	25.36	10.01	5	5
15	95.74	3.95	0	12	19.69	15.06	6	9
20	96.99	1.49	0	19	17.06	22.68	9	9
25	123.55	1.35	1	8	14.33	26.75	10	13
30	124.3	1.47	1	6	12.97	32.21	12	15
35	124.36	3.95	1	10	11.86	35.64	14	20
40	125.8	3.92	1	15	10.94	44.91	16	22

критерия оптимальности рассматривались минимальные средние затраты времени на перераспределение элементов дека между уровнями памяти в случае переполнения или опустошения быстрой памяти. В предыдущих работах в качестве критерия оптимальности рассматривалось максимальное среднее время работы до перераспределения

памяти.

В данной работе предполагалось, что вероятностные характеристики дека известны. На практике для оценки вероятностей необходимо провести эксперименты с целью вычисления частот операций с деком в конкретном приложении. Если известны функции зависимости вероятностей операций от времени, то возможны несколько вариантов нахождения оптимальных алгоритмов управления деками. Можно разбить время выполнения программы на интервалы, где вероятности операций можно считать константами, и на каждом интервале управлять деком по предложенному алгоритму. Можно искать зависимость координат оптимальной точки начала нового блуждания после перераспределения памяти (x_0, y_0) от времени.

В дальнейшем интерес представляют аналогичные задачи для стеков, FIFO-очередей, приоритетных очередей и других структур данных, а также задачи управления несколькими структурами данных в двухуровневой памяти. Можно рассматривать разные критерии оптимальности, выбор которых зависит от области применения структур данных. Если нужны очень жесткие ограничения на время решения задач, то можно, например, минимизировать не средние, а максимальные затраты на перераспределение памяти.

Список литературы

[1] M. Hasegava, Y. Shigei. “High-speed top-of-stack scheme for VLSI processor: a management algorithm and its analysis”, *Proceedings of the 12th annual international symposium on Computer architecture*, ISA’85 (June, 1985), pp. 48–54. ↑54,55

[2] T. Stanley, R. Wedig. “A performance analysis of automatically managed top of stack buffers”, *Proceedings of the 14th annual international symposium on Computer architecture*, ISCA’87 (June, 1987), pp. 272–281. ↑54,55

[3] P. J. Koopman, *Stack Computers: The New Wave*, Ellis Horwood series in computers and their applications, Halsted Press, 1989, ISBN 978-0745804187, 234 pp. ↑54

[4] А. К. Ким, В. И. Перекатов, С. Г. Ермаков. *Микропроцессоры и вычислительные комплексы семейства «Эльбрус»*, Питер, СПб., 2013, ISBN 978-5-459-01697-0, 272 с. ↑54,55

[5] Ю. В. Шевчук, А. Ю. Шевчук. «Виртуальная машина «Etherbox32vm»», *Программные системы: теория и приложения*, **7**:4(31) (2016), с. 119–143. ↑56

[6] R. D. Blumofe, C. E. Leiserson. “Scheduling multithreaded computations by work stealing”, *Journal of the ACM*, **46**:5 (1999), pp. 720–748. ↑56

[7] Д. Э. Кнут. *Искусство программирования*, пер. с англ. С. Г. Тригуб, Ю. Г. Гордиенко, И. В. Красикова и др.. Т. 1: *Основные алгоритмы*, Вильямс, М., 2002, ISBN 978-5-8459-1984-7, 720 с. ↑56,58

[8] A. V. Sokolov, E. A. Barkovsky. "The mathematical model and the problem of optimal partitioning of shared memory for work-stealing deques", PaCT 2015: Parallel Computing Technologies, Lecture Notes in Computer Science, vol. **9251**, ed. V. Malyshkin, Springer, Cham, 2015, ISBN 978-3-319-21908-0, pp. 102–106. ↑56

[9] E. A. Aksenova, A. V. Sokolov. "Modeling of the memory management process for dynamic work-stealing schedulers", 2017 Ivannikov ISPRAS Open Conference (ISPRAS) (30 Nov.–1 Dec. 2017, Moscow, Russia), 2018, pp. 12–15. ↑56

[10] А. В. Соколов, Е. А. Барковский. *Способ управления памятью компьютерной системы*, Пат. 2647627 Российская Федерация, МПК G06F9/5005 (2006.01); G06F12/02 (2006.01), заявитель и патентообладатель ФГБУН ФИЦ «Карельский научный центр Российской академии наук». № 2016143800; заявл. 08.11.2016; опубл. 16.03.2018, Бюл. № 8, 13 с. ↑56

[11] Е. А. Барковский, А. А. Лазутина, А. В. Соколов. «Построение и анализ модели процесса работы с двумя деками, двигающимися друг за другом в общей памяти», *Программные системы: теория и приложения*, **10**:1(40) (2019), с. 3–17. ↑56

[12] E. A. Aksenova, E. A. Barkovsky, A. V. Sokolov. "The models and methods of optimal control of three work-stealing deques located in a shared memory", *Lobachevskii Journal of Mathematics*, **40**:11 (2019), pp. 1763–1770. ↑56

[13] А. В. Соколов. *Математические модели и алгоритмы оптимального управления динамическими структурами данных*, Изд-во ПетрГУ, Петрозаводск, 2002, 215 с. ↑56

[14] Е. А. Барковский, А. В. Соколов. «Модель управления двумя параллельными FIFO-очередями, двигающимися друг за другом в общей памяти», *Информационно-управляющие системы*, 2016, №1, с. 65–73. ↑56

[15] R. Kuchumov, A. Sokolov, V. Korkhov. "Staccato: shared-memory work-stealing task scheduler with cache-aware memory management", *Inter. Journal of Web and Grid Services*, **15**:4 (2019), pp. 394–407. ↑56

[16] А. В. Соколов. «Об оптимальном кешировании FIFO-очередей», *Стохастическая оптимизация в информатике*, **9**:2 (2013), с. 108–123. ↑57,60

[17] E. A. Aksenova, A. A. Lazutina, A. V. Sokolov. "Study of a non-Markovian stack management model in a two-level memory", *Programming and Computer Software*, **30**:1 (2004), pp. 25–33. ↑57,60

[18] Е. А. Аксенова, А. В. Соколов. «Оптимальное управление двумя параллельными стеками в двухуровневой памяти», *Дискретная математика*, **19**:1 (2007), с. 67–75. ↑57

[19] А. А. Лазутина, А. В. Соколов. «Об оптимальном управлении Work-Stealing деками в двухуровневой памяти», *Вестник компьютерных и информационных технологий*, **17**:4 (2020), с. 51–60. ↑57

[20] А. В. Соколов. «О распределении памяти для двух стеков», *Автоматизация эксперимента и обработки данных*, КФ АН СССР, Петрозаводск, 1980, с. 65–71. ↑58

[21] A. C. Yao. "An analysis of a memory allocation scheme for implementing stacks", *SIAM J. Computing*, **10**:2 (1981), pp. 398–403. ↑58

[22] P. Flajolet. "The evolution of two stacks in bounded space and random walks in a triangle", MFCS 1986: Mathematical Foundations of Computer Science 1986, Lecture Notes in Computer Science, vol. **233**, Springer, Berlin–Heidelberg, 1986, ISBN 978-3-540-39909-4, pp. 325–340. ↑58

[23] G. Louchard, R. Schott, M. Tolley, P.Zimmermann. "Random walks, heat equation and distributed algorithms", *J. Comput. Appl. Math.*, **53**:2 (1994), pp. 243–274. ↑58

[24] G. Louchard, R. Schott. "Probabilistic analysis of some distributed algorithms", CAAP 1990: CAAP'90, Lecture Notes in Computer Science, vol. **431**, Springer, Berlin–Heidelberg, 1990, ISBN 978-3-540-52590-5, pp. 239–253. ↑58

[25] R. S. Maier. "Colliding stacks: a large deviations analysis", *Random Structures and Algorithms*, **2**:4 (1991), pp. 379–420. ↑58

[26] В. Феллер. *Введение в теорию вероятностей и ее приложения*, пер. с англ. Ю. В. Прохорова, предисл. А.Н. Колмогорова. Т. 1, Мир, М., 1984, ISBN 978-5-397-01035-1, 528 с. ↑60

[27] Т. Кохонен. *Ассоциативные запоминающие устройства*, Мир, М., 1982, 384 с. ↑62

Поступила в редакцию 15.01.2021
Переработана 09.04.2021
Опубликована 12.05.2021

Рекомендовал к публикации *к.ф.-м.н. А. В. Климов*

Пример ссылки на эту публикацию:

Е. А. Аксёнова, А. А. Лазутина, А. В. Соколов. «Минимизация средних затрат на перераспределение при работе с work-stealing деком в двухуровневой памяти». *Программные системы: теория и приложения*, 2021, **12**:2(49), с. 53–71. 10.25209/2079-3316-2021-12-2-53-71

http://psta.psiras.ru/read/psta2021_2_53-71.pdf

Об авторах:

Елена Алексеевна Аксёнова

Научный сотрудник Института прикладных математических исследований Карельского научного центра РАН. Окончила Петрозаводский государственный университет в 2004 г., к.ф.-м.н.(2007). Автор публикаций про задачи оптимального управления различными динамическими структурами данных. Область научных интересов: прикладная математика и информатика, оптимальное управление динамическими структурами данных, управляемые случайные блуждания, цепи Маркова.

0000-0002-4175-682X
e-mail: aksenova@krc.karelia.ru

Анна Александровна Лазутина

Главный специалист отдела информационных ресурсов, управления информатизации МГУ имени М.В. Ломоносова. Окончила Петрозаводский государственный университет в 2004 г., к.ф.-м.н.(2006). Автор публикаций про задачи оптимального управления стеками. Область научных интересов: прикладная математика и информатика, оптимальное управление динамическими структурами данных, управляемые случайные блуждания, цепи Маркова.

0000-0001-7569-114X
e-mail: alazutina@yandex.ru

Андрей Владимирович Соколов

Профессор, ведущий научный сотрудник Института прикладных математических исследований Карельского научного центра РАН. Окончил Ленинградский университет в 1974 г., д.ф.-м.н. (2006). Является автором более 100 научных публикаций. Область научных интересов: оптимальное управление динамическими структурами данных, оптимальное динамическое распределение нестраничной памяти, управляемые случайные блуждания, цепи Маркова, параллельные вычисления, динамические work-stealing балансировщики.

0000-0003-3787-7765
e-mail: avs@krc.karelia.ru

CSCSTI 50.07.03
UDC 004.942+004.272.3

Elena A. Aksenova, Anna A. Lazutina, Andrew V. Sokolov. *About optimal managment of work-stealing deques in two-level memory.*

Abstract. The paper analyzes the problem of optimal control of a work-stealing deque in two-level memory (for example, registers –random access memory), where probabilities of parallel operations with the deque are known. The classic sequential cyclic method for representing a deque in memory is considered. If a deque overflows or empty, we transfer elements from its middle part from the fast memory to the slow memory, since data from the end parts of the deque may be needed earlier. The problem is to find the optimal number of elements from both sides of the deque to leave in the fast memory if the deque is full or empty. As an optimality criterion, we consider the minimum average cost of memory reallocation, which is necessary in case of overflow or emptying of fast memory. The simulation model of this process is constructed. The results of numerical experiments are presented.

Key words and phrases: work-stealing balancers, work-stealing deques, Monte-Carlo method, random walks.

2020 *Mathematics Subject Classification*: 68Q85; 60J10, 68M07

References

[1] M. Hasegava, Y. Shigei. “High-speed top-of-stack scheme for VLSI processor: a management algorithm and its analysis”, *Proceedings of the 12th annual international symposium on Computer architecture*, ISA’85 (June, 1985), pp. 48–54. ↑54,55

[2] T. Stanley, R. Wedig. “A performance analysis of automatically managed top of stack buffers”, *Proceedings of the 14th annual international symposium on Computer architecture*, ISCA’87 (June, 1987), pp. 272–281. ↑54,55

[3] P. J. Koopman, *Stack Computers: The New Wave*, Ellis Horwood series in computers and their applications, Halsted Press, 1989, ISBN 978-0745804187, 234 pp. ↑54

[4] A. K. Kim, V. I. Perekatov, S. G. Yermakov. *Microprocessors and computing systems of the Elbrus family*, Piter, SPb., 2013, ISBN 978-5-459-01697-0 (in Russian), 272 pp. ↑54,55

[5] Yu. V. Shevchuk, A. Yu. Shevchuk. “Etherbox32vm virtual machine”, *Program Systems: Theory and Applications*, **7**:4(31) (2016), pp. 119–143 (in Russian). ↑56

[6] R. D. Blumofe, C. E. Leiserson. “Scheduling multithreaded computations by work stealing”, *Journal of the ACM*, **46**:5 (1999), pp. 720–748. ↑56

Supported by RFBR, project No18-01-00125-a.

10.25209/2079-3316-2021-12-2-53-71

[7] D. Knuth. *The Art of Computer Programming*. V. 1: *Fundamental Algorithms*, Addison-Wesley Professional, 1997, ISBN 978-0201896831. ↑56,58

[8] A. V. Sokolov, E. A. Barkovsky. "The mathematical model and the problem of optimal partitioning of shared memory for work-stealing deques", PaCT 2015: Parallel Computing Technologies, Lecture Notes in Computer Science, vol. **9251**, ed. V. Malyshkin, Springer, Cham, 2015, ISBN 978-3-319-21908-0, pp. 102–106. ↑56

[9] E. A. Aksenova, A. V. Sokolov. "Modeling of the memory management process for dynamic work-stealing schedulers", 2017 Ivannikov ISPRAS Open Conference (ISPRAS) (30 Nov.–1 Dec. 2017, Moscow, Russia), 2018, pp. 12–15. ↑56

[10] A. V. Sokolov, Ye. A. Barkovskiy. *Method for managing computer system memory*, Pat. 2647627 Rossiyskaya Federatsiya, MPK G06F9/5005 (2006.01); G06F12/02 (2006.01), zayavitel' i patentoobladatel' FGBUN FITs "Karel'skiy nauchnyy tsentr Rossiyskoy akademii nauk". No 2016143800; zayavl. 08.11.2016; opubl. 16.03.2018, Byul. No 8 (in Russian), 13 pp. ↑56

[11] E. A. Barkovsky, A. A. Lazutina, A. V. Sokolov. "The optimal control of two work-stealing deques, moving one after another in a shared memory", *Program Systems: Theory and Applications*, **10**:1 (2019), pp. 19–32. ↑56

[12] E. A. Aksenova, E. A. Barkovsky, A. V. Sokolov. "The models and methods of optimal control of three work-stealing deques located in a shared memory", *Lobachevskii Journal of Mathematics*, **40**:11 (2019), pp. 1763–1770. ↑56

[13] A. V. Sokolov. *Mathematical models and algorithms for optimal control of dynamic data structures*, Izd-vo PetrGU, Petrozavodsk, 2002 (in Russian), 215 pp. ↑56

[14] Ye. A. Barkovskiy, A. V. Sokolov. "Management model for two parallel FIFO queues moving one after another in shared memory", *Informatsionno-upravlyayushchiye sistemy*, 2016, no. 1, pp. 65–73 (in Russian). ↑56

[15] R. Kuchumov, A. Sokolov, V. Korkhov. "Staccato: shared-memory work-stealing task scheduler with cache-aware memory management", *Inter. Journal of Web and Grid Services*, **15**:4 (2019), pp. 394–407. ↑56

[16] A. V. Sokolov. "About the optimal caching of FIFO queues", *Stokhasticheskaya optimizatsiya v informatike*, **9**:2 (2013), pp. 108–123 (in Russian). ↑57,60

[17] E. A. Aksenova, A. A. Lazutina, A. V. Sokolov. "Study of a non-Markovian stack management model in a two-level memory", *Programming and Computer Software*, **30**:1 (2004), pp. 25–33. ↑57,60

[18] E. A. Aksenova, A. V. Sokolov. "Optimal management of two parallel stacks in two-level memory", *Discrete Math. Appl.*, **17**:1 (2007), pp. 47–55. ↑57

[19] A. A. Lazutina, A. V. Sokolov. "About optimal management of work-stealing deques in two-level memory", *Vestnik komp'yuternykh i informatsionnykh tekhnologiy*, **17**:4 (2020), pp. 51–60 (in Russian). ↑57

[20] A. V. Sokolov. "About memory allocation for two stacks", *Avtomatizatsiya eksperimenta i obrabotki dannykh*, KF AN SSSR, Petrozavodsk, 1980, pp. 65–71 (in Russian). ↑58

[21] A. C. Yao. "An analysis of a memory allocation scheme for implementing stacks", *SIAM J. Computing*, **10**:2 (1981), pp. 398–403. ↑58

[22] P. Flajolet. "The evolution of two stacks in bounded space and random walks in a triangle", MFCS 1986: Mathematical Foundations of Computer Science 1986, Lecture Notes in Computer Science, vol. **233**, Springer, Berlin–Heidelberg, 1986, ISBN 978-3-540-39909-4, pp. 325–340. ↑58

[23] G. Louchard, R. Schott, M. Tolley, P.Zimmermann. “Random walks, heat equation and distributed algorithms”, *J. Comput. Appl. Math.*, **53**:2 (1994), pp. 243–274. ↑58

[24] G. Louchard, R. Schott. “Probabilistic analysis of some distributed algorithms”, CAAP 1990: CAAP’90, Lecture Notes in Computer Science, vol. **431**, Springer, Berlin–Heidelberg, 1990, ISBN 978-3-540-52590-5, pp. 239–253. ↑58

[25] R. S. Maier. “Colliding stacks: a large deviations analysis”, *Random Structures and Algorithms*, **2**:4 (1991), pp. 379–420. ↑58

[26] W. Feller. *An Introduction to Probability Theory and its Applications*. V. I, 3rd Edition, Wiley, 1968, ISBN 978-0471257080, 509 pp. ↑60

[27] T. Kohonen, *Content-Adressable Memories*, Springer Series in Information Sciences, vol. **1**, Springer-Verlag, Berlin–Heidelberg, 1980, ISBN 978-3-642-96552-4. ↑62

Sample citation of this publication:

Elena A. Aksenova, Anna A. Lazutina, Andrew V. Sokolov. “About optimal managment of work-stealing deques in two-level memory”. *Program Systems: Theory and Applications*, 2021, **12**:2(49), pp. 53–71. (*In Russian*). 10.25209/2079-3316-2021-12-2-53-71

http://psta.psiras.ru/read/psta2021_2_53-71.pdf