ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 10**, №1(40), с. 3–17

УДК **004.942**

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

Аннотация. В work-stealing балансировщиках параллельных задач, каждое ядро имеет свой буфер задач — дек (англ. deque). Владелец дека использует один конец для добавления и извлечения задач, а из второго конца задачи перехватываются другими ядрами. В статье анализируются два метода представления деков: один из распространенных методов — раздельное последовательное циклическое представление деков; и новый предложенный нами метод, где общая память для деков заранее не делится и они двигаются друг за другом по кругу. Ранее эти методы анализировались нами для представления FIFO-очередей в сетевых приложениях, где для некоторых значений параметров системы метод «Друг за другом» давал лучший результат.

Целью исследования является построение и анализ модели процесса работы с двумя последовательными деками, когда они двигаются друг за другом по кругу в общей памяти. Математическую модель мы будем строить как случайное блуждание по целым точкам в пирамиде. Имитационная модель строится с помощью метода Монте-Карло. Используемая стратегия work-stealing — перехват одного элемента. Предложены математическая и имитационная модели данного процесса и проведены численные эксперименты.

*Ключевые слова и фразы:* work-stealing балансировщики, work-stealing деки, структуры данных, поглощающие цепи Маркова, случайные блуждания.

## Введение

Существуют динамические и статические стратегии балансировки параллельных вычислений. В статической балансировке предполагается, что известна информация о точном порядке выполнения задач. В таком случае можно построить оптимальное расписание до начала решения задач. Это удается только в редких случаях и решение в таком способе балансировки является NP-полной задачей.

Работа поддержана грантом РФФИ №18-01-00125-а.



10.25209/2079-3316-2019-10-1-3-17

В случае динамической балансировки балансировщик работает по упрощенной стратегии, где используются подходы «Work-sharing» (задачи передаются от загруженных ядер к меньше загруженным) или «Work-stealing» (пустые ядра забирают задачи у других ядер [1, 2]).

«Work-stealing» используется, например, в таких системах как Cilk [3], Cilk++, TPL [4], X10 [5], TBB, JSR166 (пакет java.util.concurrent), Erlang, OpenMP в реализации ICC [6]. Каждое ядро решает задачи, указатели на которые хранятся в деке этого ядра. Если возникает новая задача, ядро добавляет указатель на эту задачу в свой дек; когда ядру нужна задача, оно читает указатель вершины дека. Если ядро определит, что его дек пустой, оно перехватывает у другого ядра указатели на новые задачи. Перехват происходит из основания дека — как в FIFO-очереди, а операции включения и исключения выполняются как в LIFO-стеке. В работе Д. Кнута такая структура данных называется деком с ограниченным входом [7]. Можно перехватывать один элемент [1], или, например, половину элементов [8].

Деки с ограниченным входом можно представлять в памяти несколькими способами, например, в виде связных списков. В [9] была предложена модель связного представления стеков и очередей. Модель для деков будет похожа на эту модель.

Для стеков и очередей существует страничный способ представления в виде односвязного списка страниц одинаковой длины. Этот метод был предложен и проанализирован в [10]. В [11] вариант такого способа был предложен для деков, только здесь требуется уже двухсвязный список. В [12] на основе аппарата теории массового обслуживания была предложена модель work-stealing балансировщика, но в ней не рассматривался какой-либо конкретный способ представления деков в памяти.

Здесь мы будем анализировать работу двух деков. Хотя это начальный этап исследования, такая модель уже может быть важна на практике, например, для многоядерных архитектур без кэш памяти. Так, в архитектуре SEAforth у каждого ядра есть два стека (для хранения данных и адресов возврата), а в архитектуре AsAP-II у ядра есть два FIFO буфера [13]. В них очереди и стеки представлены последовательно и циклически, а при переполнении возможны потери элементов.

Работу с двумя Work-stealing деками можно реализовать аналогично, а в общем случае собирать нужные микросхемы из двудековых.

В этой статье мы предлагаем имитационную и математическую модели процесса работы с двумя параллельными work-stealing деками для нового метода работы с ними, когда деки двигаются друг за другом

по кругу в одном участке общей памяти (способ запатентован [14]). Такой метод для FIFO-очередей был описан в [15], а в [16] был проведен его анализ. Реализация дека возможна как в оперативной памяти, так и на регистрах, поэтому мы не уточняем тип памяти, в которой размещаются деки.

Математическую модель мы будем строить как случайное блуждание по целым точкам в пирамиде, где переходы осуществляет процесс в дискретном времени с заданными вероятностями, соответствующими определенным операциям с деками. Раньше такие модели мы строили для других динамических структур данных: FIFO-очередей, стеков, приоритетных очередей [9, 10, 15, 17, 18]. Результаты предварительных исследований с деками докладывались в [19].

Ранее нами были предложены и проанализированы модели для случая, когда деки представлены последовательно в разных областях памяти [20], с дискретным выполнением операций [21–23]. На основе этих моделей решались оптимизационные задачи, где в качестве критерия оптимальности рассматривалось максимальное математическое ожидание времени до перераспределения (переполнения) памяти.

Рассмотренный критерий оптимальности будет полезен в таких приложениях, где переполнение памяти является аварийной ситуацией, например, в случае применения work-stealing балансировки в приложениях реального времени или при аппаратной реализации деков [24].

Возникает вопрос о правильности использования вероятностного подхода к моделированию поведения структур данных в параллельных программах. В классических последовательных программах последовательности операций работы со структурами данных зависят от входных данных программ и определяются в реальном времени их выполнения. В параллельных программах недетерминированность еще более усиливается [25]: «*Параллельное программирование принципиально не может полностью избавиться от недетерминированности, так как соответствующие средства программирования — процессы, „потоки“ (англ. threads), их взаимодействие через общий ресурс — требуются для эффективной реализации на современной аппаратуре, а также из-за распределенного характера приложений, функционирующих в реальном мире*».

В [26] приведены данные по сбору статистики, собранной для оценки вероятностей операций работы с деками для некоторых тестов [27] (для последовательного циклического представления деков). Полученные вероятности операций использовались в численных экспериментах для анализа разработанных моделей.

Сбор статистических данных проводился с помощью версии балансировщика, где в деки записывались указатели на задачи. В этой версии для каждой задачи необходимо дважды обращаться к распределителю памяти: выделить память объекта задачи и затем освободить ее, когда задача будет завершена. Так возникают связанные с распределителем памяти накладные расходы.

В [28] описывается менеджер памяти, реализованный в этой версии экспериментального work-stealing балансировщика задач с использованием разработанных моделей [26]. В менеджере реализованы и проанализированы два метода управления памятью: *оптимальное деление* и *деление пополам*.

В описанной реализации менеджера (при использовании метода оптимального деления памяти) задачи требуют меньше памяти, но математическое ожидание времени их решения становится больше. Такие характеристики менеджера могут быть полезны в приложениях реального времени, где небольшое возрастание времени работы может быть допустимым, но переполнение памяти приводит к аварийной остановке программы.

В [29] описана новая версия балансировщика. Здесь в деках хранятся объекты, а сами деки представлены в куче с помощью двухсвязных списков массивов. Для работы с деками можно использовать те же методы и модели, что и в предыдущей версии, но в новой реализации деки будут занимать больше памяти, так как объекты задач больше чем указатели на задачи.

Обычно в work-stealing балансировщиках задач работа с деками происходит с помощью циклических массивов. Для деков из кучи (с помощью классических функций работы с памятью в трансляторах C/C++) выделяют массивы, а когда деки существенно растут или уменьшаются, то выделяются новые массивы большего или меньшего размера.

В этой работе мы решаем задачу нахождения оптимальных алгоритмов выделения памяти декам, предполагая, что известны вероятности операций, совершаемых с ними.

## 1. Математическая модель

Пусть размер общей памяти составляет $m$ единиц. В ней расположены два циклических work-stealing дека, двигающиеся по кругу, друг за другом. Пустой дек возобновляет свою работу/движение из середины свободной памяти. Схема движения структур данных показана на рисунке 1.

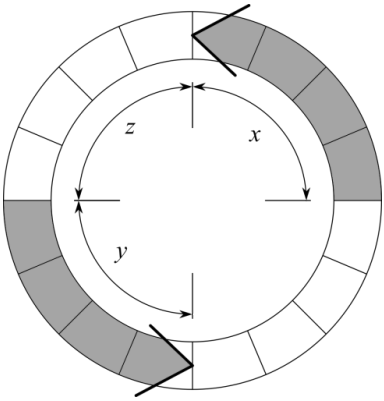


Рисунок 1. Схема движения двух work-stealing деков друг за другом по кругу

Для work-stealing деков вся работа системы разбита на операции, где за единицу времени происходит включение и/или исключение элементов.

Вероятности операций следующие:

- с вероятностью $p_1$ происходит добавление в I дек;
- с вероятностью $p_2$ происходит добавление во II дек;
- с вероятностью $p_{12}$ происходит параллельное добавление в I и II деки;
- с вероятностью $q_1$ происходит удаление из I дека;
- с вероятностью $q_2$ происходит удаление из II дека;
- с вероятностью $q_{12}$ происходит параллельное удаление из I и II деков;
- с вероятностью $pq_{12}$ происходит добавление в I дек и удаление из II дека;
- с вероятностью $pq_{21}$ происходит добавление во II дек и удаление из I дека;
- с вероятностью $r$ размер деков не изменяется (например, выполняется чтение).

$p_1 + p_2 + p_{12} + q_1 + q_2 + q_{12} + pq_{12} + pq_{21} + r = 1.$

Если система пытается исключить элемент из пустого дека, то запускается процесс work-stealing — пустой дек перехватывает работу (элементы) у другого. Была рассмотрена стратегия work-stealing — перехват одного элемента.

Требуется определить среднее время работы системы до переполнения памяти в сравнении со средним временем работы, при последовательном циклическом способе организации деков.

Обозначим текущие длины первого и второго деков как $x$ и $y$, расстояние между ними — $z$. Математической моделью процесса является случайное блуждание по целочисленной пирамиде с вершиной $(0,0,0)$ и основанием $x+y+z=m$ (рис. 2).

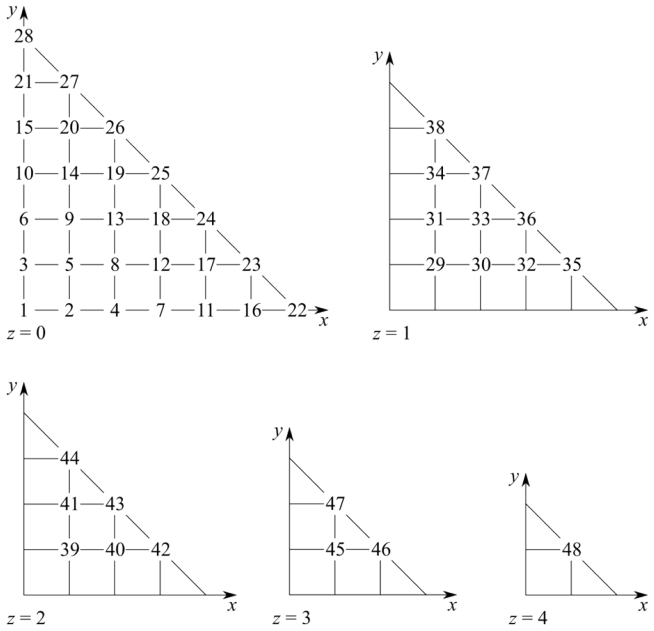


Рисунок 2. Область блуждания и нумерация состояний для двух деков в памяти размером $m=6$

Схемы переходов между состояниями приведены ниже. Здесь $(x,y,z)$ — предыдущее состояние процесса, $(x',y',z')$ — новое состояние. Блуждание по пирамиде происходит следующим образом:

$$(x,y,z) \xrightarrow{r} (x,y,z)$$

$$(x,y,z) \xrightarrow{p_1} (x',y',z')$$

$$= \begin{cases} (x+1,y,z-1), & x,y,z>0,\ x+y+z<m \\ (x+1,y,z), & 0<x<m,\ y=0,\ z=0 \\ (x+1,y,z+[\frac{m-y-1}{2}]), & x=0,\ 0<y<m,\ z=0 \end{cases}$$

$$(x,y,z) \overset{p_2}{\to} (x',y',z')$$
$$= \begin{cases} (x,y+1,z), & x,y,z>0,\ x+y+z<m \\ (x,y+1,z+[\frac{m-x-1}{2}]), & 0<x<m,\ y=0,\ z=0 \end{cases}$$

$$(x,y,z) \overset{p_{12}}{\to} (x',y',z')$$
$$= \begin{cases} (x+1,y+1,z-1), & x,y,z>0,\ x+y+z<m \\ (x+1,y+1,z+[\frac{m-x-2}{2}]), & 0<x<m-1,\ y=0,\ z=0 \\ (x+1,y+1,z+[\frac{m-y-2}{2}]), & x=0,\ 0\le y<m-1,\ z=0 \end{cases}$$

$$(x,y,z) \overset{q_1}{\to} (x',y',z')$$
$$= \begin{cases} (x-1,y,z+1), & x,y,z>0,\ x+y+z\le m \\ (x+1,y-1,z+[\frac{m-y}{2}]), & x=0,\ 1<y\le m,\ z=0 \\ (x,y,z), & x=0,\ 0\le y\le 1,\ z=0 \end{cases}$$

$$(x,y,z) \overset{q_2}{\to} (x',y',z')$$
$$= \begin{cases} (x,y-1,z), & x,y,z>0,\ x+y+z\le m \\ (x-1,y+1,z+[\frac{m-x}{2}]), & 1<x\le m,\ y=0,\ z=0 \\ (x,y,z), & 0\le x\le 1,\ y=0,\ z=0 \end{cases}$$

$$(x,y,z) \overset{q_{12}}{\to} (x',y',z')$$
$$= \begin{cases} (x-1,y-1,z+1), & x,y,z>0,\ x+y+z\le m \\ (x+1,y-2,z+[\frac{m-y}{2}]), & x=0,\ 2<y\le m,\ z=0 \\ (x-2,y+1,z+[\frac{m-x+1}{2}]), & 2\le x\le m,\ y=0,\ z=0 \\ (x-1,y,z), & 0<x\le 2,\ y=0,\ z=0 \\ (x+1,y-2,z), & x=0,\ y=2,\ z=0 \\ (x,y-1,z), & x=0,\ y=1,\ z=0 \\ (x,y,z), & x=0,\ y=0,\ z=0 \end{cases}$$

$$(x,y,z) \overset{pq_{12}}{\to} (x',y',z')$$
$$= \begin{cases} (x+1,y-1,z-1), & x,y,z>0,\ x+y+z\le m \\ (x,y+1,z+[\frac{m-x-1}{2}]), & 0<x<m,\ y=0,\ z=0 \\ (x+1,y-1,[\frac{m-y}{2}]), & x=0,\ 0<y<m,\ z=0 \\ (x+1,y,z), & x=0,\ y=0,\ z=0 \end{cases}$$

$$(x,y,z) \overset{pq_{21}}{\to} (x',y',z')$$
$$= \begin{cases} (x-1,y+1,z+1), & x,y,z>0,\ x+y+z\le m \\ (x-1,y+1,z+[\frac{m-x-1}{2}]), & 0<x<m,\ y=0,\ z=0 \\ (x+1,y,z+[\frac{m-y-1}{2}]), & x=0,\ 0<y<m,\ z=0 \\ (x,y+1,z), & x=0,\ y=0,\ z=0 \end{cases}$$

Затем, основываясь на нумерации состояний и схемах переходов, строится матрица $P$ переходных вероятностей, подматрица $Q$ которой описывает работу процесса до выхода из множества невозвратных

состояний и нужна для составления фундаментальной матрицы $N$ поглощающей цепи: $N = (I - Q)^{-1}$, где $I$ — единичная матрица. Сумма элементов матрицы $N$ в строке, соответствующей начальному состоянию, является средним временем до поглощения (переполнения), если процесс начал работу из нуля ($x = 0$ и $y = 0$) [30].

## 2. Имитационная модель

Для проверки результатов работы математической модели применялось имитационное моделирование (метод Монте-Карло). Параметрами этой модели являются размеры деков $(x, y)$, расстояние между концом первого дека и началом второго $(z)$ и вероятности операций $(p_1, p_2, p_{12}, q_1, q_2, q_{12}, pq_{12}, pq_{21}, r)$.

Для того чтобы определять какое действие будет выполняться, необходимо отрезок от 0 до 1 разделить в соответствии с вероятностями. Затем с помощью датчика случайных чисел (использовался стандартный датчик gcc) генерируется последовательность, на основе которой будет происходить блуждание по кругу (рис. 1). Для двух деков подсчитывается количество шагов, пока память не переполнится, т. е. пока $z \neq -1$ либо $x + y + z \neq m + 1$.

## 3. Некоторые примеры численного анализа

Для того чтобы проанализировать описанный в статье метод представления work-stealing деков «Друг за другом», его требуется сравнить с уже исследованным методом организации деков в памяти — последовательным циклическим, где общая память разделена пополам [26]. Для этого, над обоими методами был проведен ряд экспериментов с различным набором исходных данных.

Результаты некоторых расчетов с математическими моделями приведены в таблицах 1, 2 и 3 (указанные результаты были подтверждены имитационным исследованием). Аналитическое решение для этой задачи не известно, поэтому требуется проводить вычисления для конкретных размеров памяти $m$. Размеры $m = 4, 6, 8, 10, 100$ приводятся в качестве примера.

В таблице 1 входными данными являются теоретические вероятности — случай равных вероятностей. В таблицах 2 и 3 взяты оценки вероятностей (по частоте) возникающих при работе системы, решающей задачу перемножения матриц и задачу о рюкзаке. Эти частоты были получены в результате экспериментов с work-stealing балансировщиком задач [27]. Для этих входных значений были проведены дополнительные вычисления с имитационными моделями, где размер памяти $m = 100$ (таблицы 1, 2 и 3).

Таблица 1. Среднее время работы до переполнения ($p_1 = p_2 = p_{12} = q_1 = q_2 = q_{12} = pq_{12} = pq_{21} = 0,11$)

| $m$ | Раздельно | Друг за другом |
|---|---|---|
| 4 | 13,759 | 13,219 |
| 6 | 21,208 | 22,872 |
| 8 | 31,888 | 33,595 |
| 10 | 45,508 | 45,143 |
| 100 | 3158,0 | 3297,0 |

Таблица 2. Среднее время работы до переполнения, задача перемножения матриц ($p_1 = 0,071$, $p_2 = 0,108$, $p_{12} = 0,014$, $q_1 = 0,071$, $q_2 = 0,108$, $q_{12} = 0,014$, $pq_{12} = 0,013$, $pq_{21} = 0,013$)

| $m$ | Раздельно | Друг за другом |
|---|---|---|
| 4 | 37,050 | 38,932 |
| 6 | 57,243 | 65,721 |
| 8 | 88,169 | 88,458 |
| 10 | 126,268 | 119,246 |
| 100 | 8878,300 | 9320,350 |

Таблица 3. Сравнение времени работы до переполнения, задача о рюкзаке ($p_1 = 0,025$, $p_2 = 0,05$, $p_{12} = 0,002$, $q_1 = 0,025, q_2 = 0,05, q_{12} = 0,002, pq_{12} = 0,002, pq_{21} = 0,002$)

| $m$ | Раздельно | Друг за другом |
|---|---|---|
| 4 | 101,310 | 112,718 |
| 6 | 156,527 | 187,609 |
| 8 | 242,151 | 241,730 |
| 10 | 346,735 | 329,609 |
| 100 | 24334,267 | 26129,358 |

Время работы системы, реализованной по способу «Друг за другом», сравнивается со временем работы системы, в которой общая память заранее разделена между деками: каждому деку выделено $m/2$ единиц памяти. Стратегия work-stealing: перехват одного элемента.

Анализируя представленные результаты можно сделать вывод, что для некоторых размеров памяти система, построенная на основе предлагаемого способа, работает дольше. В частности, для задачи

перемножения матрицы разница во времени работы системы до переполнения (при памяти размером $m = 6$) составляет $8,5$, то есть система работает на 8 операций дольше, если деки расположены в общей памяти друг за другом по кругу. А для задачи о рюкзаке — на 31 операцию дольше.

С увеличением размера памяти, разница во времени работы системы также увеличивается. Так, при $m = 100$ для задачи перемножения матрицы система, где деки расположены друг за другом, работает на 442 операций дольше, а для задачи о рюкзаке — уже на 1795 операций.

## 4. Заключение

Были построены математическая и имитационная модели процесса работы двух work-stealing деков, двигающихся друг за другом. Был проведен численный анализ этого метода представления деков в памяти, где эксперименты основывались на практических и теоретических данных.

Математическая модель была представлена в виде случайного блуждания по целочисленной пирамиде с отражающими и поглощающими экранами. Разработаны алгоритм и программа вычисления среднего времени работы системы до переполнения. Разработана имитационная модель. Сам метод представления деков друг за другом по кругу был запатентован [14].

Для каждой задачи проводилось 10 млн имитационных экспериментов, при меньшем количестве были расхождения с результатами математического моделирования. Такое количество экспериментов можно объяснить большой дисперсией случайной величины (числа шагов до переполнения), но стоит заметить, что в этих задачах она не достаточно точно определяет качество экспериментов. Так как здесь максимизируется среднее время работы, любое отклонение от среднего значения вверх является желательным.

Также можно отметить, что в статье не дается классической постановки задачи оптимизации. Здесь критерий оптимальности невозможно записать аналитически, и он вычисляется алгоритмически. Решить систему разностных уравнений в этой задаче не удастся, но в некоторых других задачах такие системы решались численно [32].

Предложенные модель, алгоритм и программа для анализа метода движения деков «друг за другом» можно использовать при реализации операционных систем, планировщиков, менеджеров памяти и других системных программ.

С помощью разработанной модели можно, при заданных вероятностных характеристиках деков, выбрать лучший метод представления

структур данных, например, из двух методов: классический последовательный циклический метод, или метод «друг за другом».

В [17] и [31] были предложены математические модели для оптимального управления одним и двумя стеками в двухуровневой памяти, а в [32] — для управления FIFO-очередями. На практике в разных архитектурах был аппаратно реализован ряд методов управления стеками в двухуровневой памяти, как альтернатива классической кэш-памяти [33]. В статье мы рассмотрели модель для оптимального управления двумя деками в памяти одного уровня, но в будущем будет важно построить модели для оптимального управления деками в двухуровневой памяти.

Наш опыт программной реализации показал большое значение правильного использования кэш-памяти в параллельных балансировщиках задач, работающих по стратегии work-stealing. Например, в реализации [29] за счет этого удалось уменьшить накладные расходы планировщика в 2,5 раза и промахи на последнем уровне кэша на 30% по сравнению с work-stealing планировщиками Intel TBB и Intel/MIT Cilk.

Поэтому нужно исследовать и реализовать оптимальные аппаратные реализации деков, а не пытаться подстраиваться под универсальные реализации кэш-памяти.

## Список литературы


[1] R.D. Blumofe, C.E. Leiserson. “Scheduling multithreaded computations by work stealing”, *Journal of the ACM*, **5**:46 (1999), pp. 720–748. ↑4

[2] M. Herlihy, N. Shavit. *The art of multiprocessor programming*, Elsevier, 2008, 536 pp. ↑4

[3] R.D. Blumofe, C.F. Joerg, B.C. Kuszmaul, C.E. Leiserson, K.H. Randall, Y. Zhou. “Cilk: an efficient multithreaded runtime system”, *Journal of Parallel and Distributed Computing*, **37**:1 (1996), pp. 55–69. ↑4

[4] D. Leijen, W. Schulte, S. Burckhardt. “The design of a task parallel library”, *ACM SIGPLAN*, **44**:10 (2009), pp. 227–242. ↑4

[5] O. Tardieu, H. Wang, H. Lin. “A work-stealing scheduler for X10’s task parallelism with suspension”, *ACM SIGPLAN*, **47**:8 (2012), pp. 267–276. ↑4

[6] G. Varisteas. *Effective cooperative scheduling of task-parallel applications on multiprogrammed parallel architectures*, Doctoral Thesis in Information and Communication Technology, Royal institute of Technology, KTH, Stockholm, Sweden, 2015. ↑4

[7] D. Knuth. *The art of multiprocessor programming*. V. 1, Addison-Wesley Professional, 1997, 672 pp. ↑4

[8] D. Hendler, N. Shavit. “Non-blocking steal-half work queues”, *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, PODC ’02, 2002, pp. 280–289. ↑4

[9] A.V. Sokolov, A.V. Drac. “The linked list representation of $n$ LIFO-stacks and/or FIFO-queues in the single-level memory”, *Information Processing Letters*, **113**:19–21 (2013), pp. 832–835. ↑4,5

[10] Е.А. Аксенова, А.А. Лазутина, А.В. Соколов. «Об оптимальных методах представления динамических структур данных», *Обозрение прикладной и промышленной математики*, **10**:2 (2003), с. 375–376. ↑4,5

[11] D. Hendler, Y. Lev, M. Moir, N. Shavit. “A dynamic-sized nonblocking work stealing deque”, *Distributed Computing*, **18**:3 (2006), pp. 189–207. ↑4

[12] M. Mitzenmacher. “Analyses of load stealing models based on differential equations”, *Proceedings of the tenth annual ACM symposium on Parallel algorithms and architectures*, SPAA ’98, 1998, pp. 212–221. ↑4

[13] А.В. Калачев. *Многоядерные процессоры*, БИНОМ, М., 2014, 247 с. ↑4

[14] Е.А. Барковский, А.В. Соколов. *Способ управления памятью компьютерной системы*, № 2647627. Бюллетень № 8, Опубл. 16.03.2018. ↑5,12

[15] А.В. Соколов. *Математические модели и алгоритмы оптимального управления динамическими структурами данных*, ПетрГУ, Петрозаводск, 2002. ↑5

[16] 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. ↑5

[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. ↑5,13

[18] E.A. Aksenova, A.V. Sokolov. “Optimal implementation of two FIFO-queues in single-level memory”, *Applied Mathematics*, **2**:10 (2011), pp. 1297–1302. ↑5

[19] Е.А. Барковский, А.А. Лазутина, А.В. Соколов. «Модель управления двумя work-stealing деками, двигающимися друг за другом в общей памяти», *Обозрение прикладной и промышленной математики*, **25**:1 (2018), с. 77. ↑5

[20] D. Chase, Y. Lev. “Dynamic circular work-stealing deque”, *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, SPAA ’05, 2005, pp. 21–28. ↑5

[21] Е.А. Барковский, А.В. Соколов. «Вероятностная модель для задачи оптимального управления work-stealing деками при различных стратегиях перехвата работы», *Вероятностные методы в дискретной математике*, IX Международная Петрозаводская конференция, 2016, с. 11–13. ↑5

[22] 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**, 2015, pp. 102–106. ↑5

[23] E.A. Aksenova, A.V. Sokolov. “Modeling of the memory management process for dynamic work-stealing schedulers”, 2017 Ivannikov ISPRAS Open Conference (ISPRAS), 2018, pp. 12–15. ↑5

[24] S. Mattheis, T. Schuele, A. Raabe, T. Henties, U. Gleim. “Work stealing strategies for parallel stream processing in soft real-time systems”, ARCS 2012: Architecture of Computing Systems – ARCS 2012, Lecture Notes in Computer Science, vol. **7179**, 2002, pp. 172–183. ↑5

[25] А.И. Адамович, А.В. Климов. «Как создавать параллельные программы, детерминированные по построению? Постановка проблемы и обзор работ», *Программные системы: теория и приложения*, **8**:4 (2017), с. 221–244. ↑5

[26] Е.А. Барковский, Р.И. Кучумов, А.В. Соколов. «Оптимальное управление двумя work-stealing деками в общей памяти при различных стратегиях перехвата работы», *Программные системы: теория и приложения*, **8**:1 (2017), с. 83–103. ↑5,6,10

[27] Р.И. Кучумов. «Реализация и анализ work-stealing планировщика задач», *Стохастическая оптимизация в информатике*, **12**:1 (2016), с. 20–39. ↑5,10

[28] Е.А. Барковский. «Реализация менеджера памяти в work-stealing балансировщике», *Стохастическая оптимизация в информатике*, **13**:1 (2017), с. 56–65. ↑6

[29] R. Kuchumov, A. Sokolov, V. Korkhov. “Staccato: cache-aware work-stealing task scheduler for shared-memory systems”, ICCSA 2018: Computational Science and Its Applications – ICCSA 2018, Lecture Notes in Computer Science, vol. **10963**, 2018, pp. 91–102. ↑6,13

[30] Дж. Кемени, Дж. Снелл. *Конечные цепи Маркова*, Наука, М., 1970, 272 с. ↑10

[31] Е.А. Аксенова, А.В. Соколов. «Оптимальное управление двумя параллельными стеками в двухуровневой памяти», *Дискретная математика*, **19**:1 (2007), с. 67–75. ↑13

[32] А.В. Соколов. «Об оптимальном кешировании FIFO-очередей», *Стохастическая оптимизация в информатике*, **9**:2 (2013), с. 108–123. ↑12,13

[33] P.J. Koopman. *Stack computers: the new wave*, Ellis Horwood Ltd., 1989, 502 pp. ↑13

*Поступила в редакцию* 28.10.2018
*Переработана* 20.11.2018
*Опубликована* 18.02.2019

Рекомендовал к публикации *к.т.н. С. А. Амелькин*

*Об авторах:*

**Евгений Александрович Барковский**

Окончил Петрозаводский государственный университет в 2012 г. В 2015 г. закончил аспирантуру Института прикладных математических исследований Карельского научного центра РАН (лаборатория информационных компьютерных технологий). Автор 17 научных публикаций, посвященных параллельным динамическим структурам данных. Область научных интересов: задачи оптимального управления параллельными динамическими структурами данных, work-stealing балансировщики.

0000-0001-9041-6453
**e-mail:** barkevgen@gmail.com

**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:** sokavs@gmail.com

*Пример ссылки на эту публикацию:*

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*, 2019, **10**:1(40), с. 3–17. 10.25209/2079-3316-2019-10-1-3-17
http://psta.psiras.ru/read/psta2019_1_3-17.pdf
Эта же статья по-английски: 10.25209/2079-3316-2019-10-1-19-32

*Sample citation of this publication:*

Eugene Barkovsky, Anna Lazutina, Andrew Sokolov. “The Optimal Control of Two Work-Stealing Deques, Moving One After Another in a Shared Memory”. *Program Systems: Theory and Applications*, 2019, **10**:1(40), pp. 3–17. (*In Russian*). 10.25209/2079-3316-2019-10-1-3-17
http://psta.psiras.ru/read/psta2019_1_3-17.pdf
The same article in English: 10.25209/2079-3316-2019-10-1-19-32