УДК 004.942 **DOI: 10.14529/cmse240303**

ОПТИМАЛЬНОЕ УПРАВЛЕНИЕ ТРЕМЯ WORK-STEALING ДЕКАМИ В ДВУХУРОВНЕВОЙ ПАМЯТИ

© 2024 Е.А. Аксёнова, Е.А. Барковский, А.В. Соколов

Институт прикладных математических исследований
Карельского научного центра Российской академии наук
(185910 Петрозаводск, ул. Пушкинская, д. 11)
E-mail: aksenova@krc.karelia.ru, barkevgen@gmail.com, avs@krc.karelia.ru

Поступила в редакцию: 26.07.2024

При выполнении параллельных вычислений возникает проблема равномерного разделения задач между потоками. Одним из способов решения этой проблемы является применение распределенной динамической балансировки нагрузки. При таком способе балансировки каждый рабочий поток имеет свою очередь задач и потоки сами занимаются дальнейшем распределением задач. Широкое распространение получил метод балансировки «work-stealing», в котором один поток, у которого закончились задачи, может перехватывать задачи других потоков. Для реализации такого метода у каждого потока должен быть свой специализированный дек, в котором хранятся указатели на задачи. В этой статье предлагается и исследуется новый метод представления трех work-stealing деков в двухуровневой памяти. Рассматривается случай работы с тремя деками, когда в одном разделе быстрой памяти расположены две LIFO-части деков — два стека, которые растут навстречу друг другу, в другом разделе быстрой памяти расположены три FIFO-части, которые объединены в одну FIFO-очередь, из которой элементы только исключаются (кражи), и третья LIFO-часть. Средние части деков находятся в медленной памяти, обращение к ним происходит при переполнении или опустошении LIFO-частей или опустошении FIFO-частей деков, расположенных в быстрой памяти. Рассматривается задача оптимального разделения быстрой памяти для трех деков с заранее заданными вероятностями выполнения операций. Этот выбор зависит от характеристик уровней памяти, вероятностей операций и критерия оптимальности. В качестве критерия оптимальности рассматривается максимальное среднее время работы системы (среднее количество операций) до перераспределения памяти.

Ключевые слова: двухуровневая память, метод Монте-Карло, структуры данных, work-stealing деки.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Аксёнова Е.А., Барковский Е.А., Соколов А.В. Оптимальное управление тремя work-stealing деками в двухуровневой памяти // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 3. С. 47–60. DOI: 10.14529/cmse240303.

Введение

При выполнении параллельных вычислений возникает проблема равномерного разделения задач между потоками. Эту проблему решают с помощью статической и динамической балансировки задач [1]. Если свойства и особенности вычисляемых задач известны заранее, то применяют статическую балансировку. Такие задачи являются NP-полными [2], и для их решения можно составить расписание. Если информации о вычисляемых задач недостаточно, то целесообразно применять динамическую балансировку. В таком случае балансировка задач происходит во время работы системы [3]. Стратегию динамической балансировки, в свою очередь, можно разделить на централизованную и распределенную [4, 5]. В централизованной балансировке назначением задач занят специально для этого отведенный поток.

В распределенной балансировке каждый рабочий поток имеет свою очередь задач и потоки сами занимаются дальнейшем распределением задач. Существуют различные ме-

тоды распределенной балансировки задач. По методу «work-dealing» поток, у которого накопилось много задач, перераспределяет их среди других потоков [6, 7]. По методу «work-requesting» поток, у которого заканчиваются задачи, запрашивает их у других потоков [8]. Суть метода «work-stealing» в том, что если у потока заканчиваются задачи, он начинает перехватывать их у других потоков [9]. Метод work-stealing имеет широкое распространение [10] и применяется в следующих балансировщиках задач: Cilk; Intel TBB; dotNET TPL; X10 и других. Для реализации этого метода, у каждого потока должен быть свой специализированный дек, в котором хранятся указатели на задачи. Дек (англ. «double ended queue», сокращенно «deque») — это структура данных, в которой операции добавления (операция «push») и удаления (операция «pop») элементов происходят с обоих концов по принципу LIFO (рис. 1).

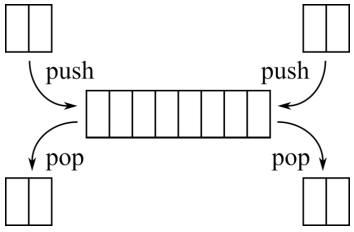

Рис. 1. Принцип работы дека

Во время работы системы, реализованной на основе метода work-stealing, потоки обращаются к своим декам по следующему принципу: после создания новой задачи, поток добавляет указатель на нее в вершину своего дека; если потоку требуется задача для выполнения и его дек не пуст, он берет указатель из вершины дека; если потоку требуется задача для выполнения и его дек пуст, он перехватывает указатель из основания дека другого потока. Таким образом, первые две операции выполняются с одного конца по принципу LIFO, а перехват элементов происходит из основания структуры данных (принцип FIFO). Дек, работающий таким образом, называется деком с ограниченным входом [11] или work-stealing деком (рис. 2).

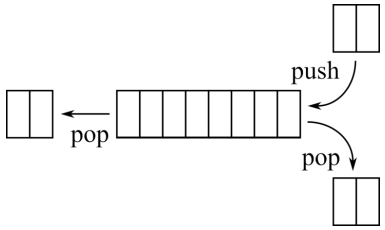

Рис. 2. Принцип работы work-stealing дека

Так как задачи балансировки нагрузки возникают в многопроцессорных системах [12] и в вычислительных сетях [13, 14], требуется эффективно использовать метод work-stealing. Для этого предлагают не только различные способы его реализации [15], но и способы управления work-stealing деками [16, 17].

Модель work-stealing балансировщика, основанная на аппарате теории массового обслуживания, была предложена в [18]. В [19, 20] предлагается и исследуется реализация экспериментального динамического work-stealing балансировщика. Деки можно представить в памяти с помощью различных методов [21]. Например, метод связного представления [22].

Для стеков и очередей модель на основе этого метода уже построена [23]. Также можно воспользоваться методом двухсвязного страничного представления [24]. Модель на основе этого метода для деков будет похожа на уже построенную модель для других структур данных [25].

В [26, 27] предлагались модели и методы оптимального управления двумя work-stealing деками в общей памяти одного уровня. Было рассмотрено использование распространенного метода раздельного последовательного циклического представления и применение нового метода, где деки двигаются друг за другом по кругу [28]. В [29] были предложены и проанализированы модели и методы оптимального управления тремя work-stealing деками в общей памяти одного уровня.

В двухуровневой памяти дек представлен таким образом: в быстрой памяти первого уровня расположены концы дека с которыми будут происходить операции; в медленной памяти второго уровня находится середина дека (рис. 3). В [30, 31] предлагались модели работы дека в двухуровневой памяти и решалась задача о поиске оптимального числа элементов концов дека, которые остаются в быстрой части после перераспределения двухуровневой памяти. Задача оптимального управления двумя work-stealing деками в двухуровневой памяти была исследована в [32].

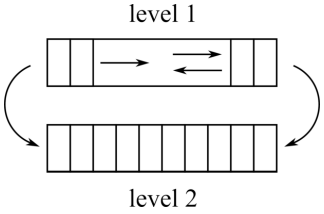

Рис. 3. Принцип работы work-stealing дека в двухуровневой памяти

В этой статье предлагается и исследуется новый метод представления трех work-stealing деков в двухуровневой памяти. В разделе 1 описан новый метод представления трех деков в двухуровневой памяти и сформулирована задача оптимального управления деками. Решение задачи и результаты исследований даны в разделе 2. В заключении предложен вариант использования этого метода и приведены некоторые направления дальнейших исследований.

1. Постановка задачи

Рассмотрим компьютерную систему, имеющую двухуровневую память. В медленной памяти второго уровня расположены середины трех деков (De_1, De_2, De_3), в которых хранятся указатели на задачи. В быстрой памяти первого уровня расположены концы (Sk_n, Qe_n) трех деков (De_n): Sk_n — рабочий конец дека De_n, в который происходят добавления и удаления указателей; Qe_n — work-stealing конец дека De_n, из которого посторонние (не рассматриваемые здесь) деки перехватывают указатели.

Память первого уровня размером m единиц разделена на две части s и $m - s$ (рис. 4): в части s расположены work-stealing концы всех трех деков (Qe_1, Qe_2, Qe_3), объединенные в одну очередь Qe и рабочий конец первого дека Sk_1, работающий как стек; в части $m - s$ расположены рабочие концы остальных двух деков (Sk_2, Sk_3), растущие навстречу друг другу.

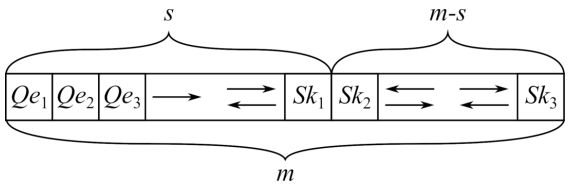

Рис. 4. Расположение деков в памяти первого уровня

Один указатель на задачу равен одной единице памяти. Пусть начальные размеры концов деков $Sk_1 = Qe_1 = Sk_2 = Qe_2 = Sk_3 = Qe_3 = 10$ указателей, общий размер памяти первого уровня $m = 100$ указателей. Таким образом, разделение s может принимать значения $s = 40 \dots 80$ единиц памяти. Начальные размеры концов деков и размер памяти являются теоретическими значениями, взятыми для упрощения дальнейших расчетов. В этой статье не рассматривается задача нахождения оптимальных начальных размеров. В табл. 1 приведены операции, происходящие с каждым деком на каждом шаге дискретного времени с заданной вероятностью. Сумма вероятностей возникновения операций для дека De_n: $p_n + q_n + w_n + pw_n + qw_n + r_n = 1$.

Таблица 1. Вероятности операций с деками

Обозначение	Описание вероятности
p_n	Добавление указателя в рабочий конец Sk_n дека De_n
q_n	Удаление указателя из рабочего конца Sk_n дека De_n
w_n	Перехват указателя из общей work-stealing очереди Qe, в то время как дек De_n обрабатывает задачу
pw_n	Параллельное добавление указателя в рабочий конец Sk_n дека De_n и перехват указателя из общей work-stealing очереди Qe
qw_n	Параллельное удаление указателя из рабочего конца Sk_n дека De_n и перехват указателя из общей work-stealing очереди Qe
r_n	Операция не изменяющая размеры концов дека De_n (например, обработка задачи)

Система работает бесконечно и прекращает свою работу (происходит перераспределение быстрой памяти) при опустошении рабочего конца любого дека $Sk_n < 0$, при опустошении общей work-stealing очереди Qe или при переполнении любой части быстрой памяти $Qe + Sk_1 > s$, $Sk_2 + Sk_3 > (m - s)$. В этой статье решается задача о нахождении оптимального значения разделения s быстрой памяти m для трех деков с заранее заданными вероятностями выполнения операций. Критерий оптимальности: максимальное среднее время работы системы (среднее количество операций) до перераспределения памяти.

2. Численный анализ

Для решения поставленной задачи была разработана имитационная модель процесса на основе метода Монте-Карло. Входными данными модели значатся: размер памяти первого уровня (m); часть памяти отведенная паре Qe—Sk_1 (s); вероятности возникновения операций с деками (p_n, q_n, w_n, pw_n, qw_n, r_n). Результатом имитационного моделирования является среднее время работы системы (t) до перераспределения памяти. С помощью имитационной модели был проведен ряд исследований. Взятые здесь вероятности операций являются теоретическими, для упрощения расчетов и повышения наглядности результатов.

В табл. 2 и 3 рассматривается ситуации, в которых указатели добавляются в один из деков чаще, чем в остальные деки ($p_1 > p_2$, $p_1 > p_3$ и $p_2 > p_1$, $p_2 > p_3$). В табл. 4 и 5 анализируются случаи, где указатели добавляются в два дека чаще, чем в оставшийся дек ($p_1 > p_3$, $p_2 > p_3$ и $p_2 > p_1$, $p_3 > p_1$). В табл. 6 рассматривается случай, где указатели добавляются во все деки одинаково часто ($p_1 = p_2 = p_3$).

Таблица 2. Среднее время работы системы, $p_1 > p_2$ и p_3

Вероятности операций	Метод разделения памяти	
	Разделение пополам	Оптимальное разделение
$p_1 = 0.50$, $p_2 = p_3 = 0.26$, $q_1 = 0.02$, $q_2 = q_3 = 0.26$, $r_1 = r_2 = r_3 = 0.45$	27.90	66.72 ($s = 69$)
$p_1 = 0.60$, $p_2 = p_3 = 0.31$, $q_1 = 0.02$, $q_2 = q_3 = 0.31$, $r_1 = r_2 = r_3 = 0.35$	22.24	54.09 ($s = 70$)
$p_1 = 0.70$, $p_2 = p_3 = 0.36$, $q_1 = 0.02$, $q_2 = q_3 = 0.36$, $r_1 = r_2 = r_3 = 0.25$	18.49	45.55 ($s = 70$)
$p_1 = 0.80$, $p_2 = p_3 = 0.41$, $q_1 = 0.02$, $q_2 = q_3 = 0.41$, $r_1 = r_2 = r_3 = 0.15$	15.82	39.34 ($s = 70$)
$p_1 = 0.90$, $p_2 = p_3 = 0.46$, $q_1 = 0.02$, $q_2 = q_3 = 0.46$, $r_1 = r_2 = r_3 = 0.05$	13.83	34.63 ($s = 70$)

Таблица 3. Среднее время работы системы, $p_2 > p_1$ и p_3

Вероятности операций	Метод разделения памяти	
	Разделение пополам	Оптимальное разделение
$p_2 = 0.50$, $p_1 = p_3 = 0.26$, $q_2 = 0.02$, $q_1 = q_3 = 0.26$, $r_1 = r_2 = r_3 = 0.45$	60.16	64.55 ($s = 46$)
$p_2 = 0.60$, $p_1 = p_3 = 0.31$, $q_2 = 0.02$, $q_1 = q_3 = 0.31$, $r_1 = r_2 = r_3 = 0.35$	49.94	53.29 ($s = 46$)
$p_2 = 0.70$, $p_1 = p_3 = 0.36$, $q_2 = 0.02$, $q_1 = q_3 = 0.36$, $r_1 = r_2 = r_3 = 0.25$	42.69	45.36 ($s = 46$)
$p_2 = 0.80$, $p_1 = p_3 = 0.41$, $q_2 = 0.02$, $q_1 = q_3 = 0.41$, $r_1 = r_2 = r_3 = 0.15$	37.29	39.48 ($s = 46$)
$p_2 = 0.90$, $p_1 = p_3 = 0.46$, $q_2 = 0.02$, $q_1 = q_3 = 0.46$, $r_1 = r_2 = r_3 = 0.05$	33.10	34.94 ($s = 46$)

Таблица 4. Среднее время работы системы, p_1 и $p_2 > p_3$

Вероятности операций	Метод разделения памяти	
	Разделение пополам	Оптимальное разделение
$p_1 = p_2 = 0.50$, $p_3 = 0.26$, $q_1 = q_2 = 0.02$, $q_3 = 0.26$, $r_1 = r_2 = r_3 = 0.45$	27.86	41.01 ($s = 58$)
$p_1 = p_2 = 0.60$, $p_3 = 0.31$, $q_1 = q_2 = 0.02$, $q_3 = 0.31$, $r_1 = r_2 = r_3 = 0.35$	22.24	33.73 ($s = 58$)
$p_1 = p_2 = 0.70$, $p_3 = 0.36$, $q_1 = q_2 = 0.02$, $q_3 = 0.36$, $r_1 = r_2 = r_3 = 0.25$	18.49	28.75 ($s = 59$)
$p_1 = p_2 = 0.80$, $p_3 = 0.41$, $q_1 = q_2 = 0.02$, $q_3 = 0.41$, $r_1 = r_2 = r_3 = 0.15$	15.83	25.12 ($s = 59$)
$p_1 = p_2 = 0.90$, $p_3 = 0.46$, $q_1 = q_2 = 0.02$, $q_3 = 0.46$, $r_1 = r_2 = r_3 = 0.05$	13.83	22.35 ($s = 59$)

Таблица 5. Среднее время работы системы, p_2 и $p_3 > p_1$

Вероятности операций	Метод разделения памяти	
	Разделение пополам	Оптимальное разделение
$p_2 = p_3 = 0.50$, $p_1 = 0.26$, $q_2 = q_3 = 0.02$, $q_1 = 0.26$, $r_1 = r_2 = r_3 = 0.45$	32.47	36.45 ($s = 45$)
$p_2 = p_3 = 0.60$, $p_1 = 0.31$, $q_2 = q_3 = 0.02$, $q_1 = 0.31$, $r_1 = r_2 = r_3 = 0.35$	26.92	30.13 ($s = 45$)
$p_2 = p_3 = 0.70$, $p_1 = 0.36$, $q_2 = q_3 = 0.02$, $q_1 = 0.36$, $r_1 = r_2 = r_3 = 0.25$	23.01	25.69 ($s = 45$)
$p_2 = p_3 = 0.80$, $p_1 = 0.41$, $q_2 = q_3 = 0.02$, $q_1 = 0.41$, $r_1 = r_2 = r_3 = 0.15$	20.09	22.39 ($s = 45$)
$p_2 = p_3 = 0.90$, $p_1 = 0.46$, $q_2 = q_3 = 0.02$, $q_1 = 0.46$, $r_1 = r_2 = r_3 = 0.05$	17.85	19.86 ($s = 45$)

Таблица 6. Среднее время работы системы, $p_1 = p_2 = p_3$

Вероятности операций	Метод разделения памяти	
	Разделение пополам	Оптимальное разделение
$p_1 = p_2 = p_3 = 0.50$, $q_1 = q_2 = q_3 = 0.02$, $r_1 = r_2 = r_3 = 0.45$	25.87	27.90 ($s = 53$)
$p_1 = p_2 = p_3 = 0.60$, $q_1 = q_2 = q_3 = 0.02$, $r_1 = r_2 = r_3 = 0.35$	21.14	23.17 ($s = 53$)
$p_1 = p_2 = p_3 = 0.70$, $q_1 = q_2 = q_3 = 0.02$, $r_1 = r_2 = r_3 = 0.25$	17.88	19.88 ($s = 53$)
$p_1 = p_2 = p_3 = 0.80$, $q_1 = q_2 = q_3 = 0.02$, $r_1 = r_2 = r_3 = 0.15$	15.49	17.49 ($s = 53$)
$p_1 = p_2 = p_3 = 0.90$, $q_1 = q_2 = q_3 = 0.02$, $r_1 = r_2 = r_3 = 0.05$	13.67	15.68 ($s = 53$)

На основе проведенных исследований можно сделать вывод, что среднее время работы системы возрастает, если делить память оптимально. Например, по результатам табл. 2 для общей памяти $m = 100$ единиц и вероятностях $p_1 = 0.50$, $p_2 = p_3 = 0.26$, оптимальные размеры памяти составляют: $s = 69$ единиц для пары Qe—Sk_1 и $m - s = 31$ единиц для пары Sk_2—Sk_3. При таком разделении памяти система в среднем совершает на 38.82 операций больше, чем при разделении памяти пополам. График зависимости среднего времени работы (t) от разделения памяти (s) приведен на рис. 5.

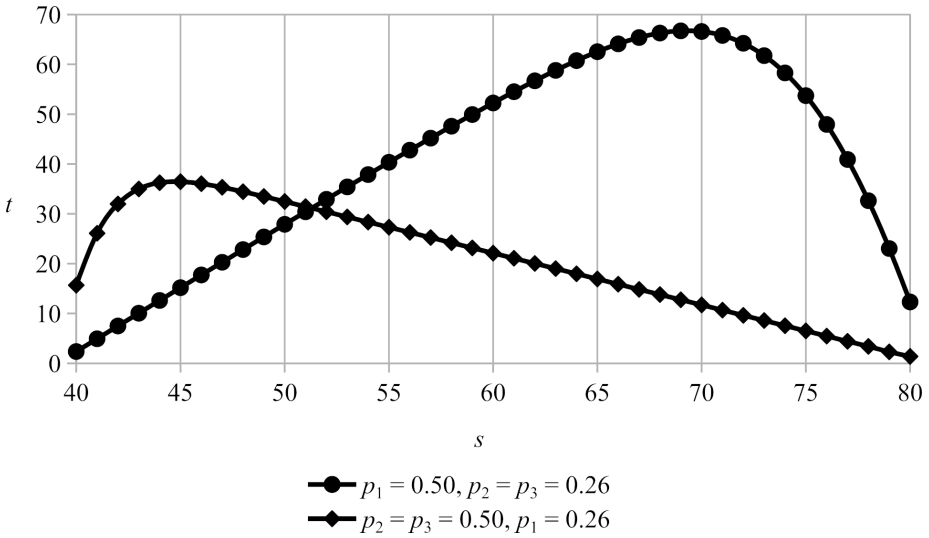

Рис. 5. Зависимость времени работы от разделения памяти

Аналогичная ситуация наблюдается и для остальных случаев. Так, анализируя табл. 5 можно увидеть, что оптимальные размеры памяти для вероятностей $p_2 = p_3 = 0.50$,

$p_1 = 0.26$ составляют: $s = 45$ единиц и $m - s = 55$ единиц. При оптимальном разделении памяти система в среднем совершает 36.45 операций, что на 3.98 операций больше, чем при разделении памяти пополам. График зависимости среднего времени работы (t) от разделения памяти (s) приведен на рис. 5.

Заключение

В статье решается задача о поиске оптимального значения разделения памяти первого уровня между тремя work-stealing деками, в качестве критерия оптимальности рассматривалось максимальное среднее время работы системы до перераспределения памяти. Имитационная модель этого процесса была построена на основе метода Монте-Карло и проанализирована.

Предложенную модель можно использовать для оптимизации двухуровневой памяти при разработке системного программного обеспечения. Для этого требуется провести предварительный статистический анализ работы системы и на основе полученных данных рассчитать значение оптимального разделения памяти.

В будущем можно рассмотреть другие критерии оптимальности, такие как минимизация суммы средних затрат или минимизация наибольших средних затрат на перераспределение быстрой памяти. Также можно обобщить эту задачу на произвольное число деков. В таком случае надо будет рассматривать разные варианты совместного расположения LIFO- и FIFO-частей деков в быстрой памяти.

Литература

1. Herlihy M., Shavit N. The Art of Multiprocessor Programming. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008. DOI: 10.5555/1734069.
2. Kwok Y.-K., Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors // ACM Comput. Surv. New York, NY, USA, 1999. Dec. Vol. 31, no. 4. P. 406–471. DOI: 10.1145/344588.344618.
3. Alakeel A.M. A Guide to Dynamic Load Balancing in Distributed Computer Systems // International Journal of Computer Science and Network Security. 2010. Vol. 10, no. 6. P. 153–160.
4. Beaumont O., Carter L., Ferrante J., *et al.* Centralized versus distributed schedulers for multiple bag-of-task applications // Proceedings 20th IEEE International Parallel & Distributed Processing Symposium. 2006. P. 10. DOI: 10.1109/IPDPS.2006.1639262.
5. Xia Y., Prasanna V.K., Li J. Hierarchical Scheduling of DAG Structured Computations on Manycore Processors with Dynamic Thread Grouping // Job Scheduling Strategies for Parallel Processing / ed. by E. Frachtenberg, U. Schwiegelshohn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 154–174. DOI: 10.1007/978-3-642-16505-4_9.
6. Hendler D., Shavit N. Non-blocking steal-half work queues // Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing. Monterey, California: Association for Computing Machinery, 2002. P. 280–289. DOI: 10.1145/571825.571876.
7. Hendler D., Shavit N. Work dealing // Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures. Winnipeg, Manitoba, Canada: Association for Computing Machinery, 2002. P. 164–172. DOI: 10.1145/564870.564900.

8. Acar U.A., Charguéraud A., Rainey M. Scheduling parallel programs by work stealing with private deques // SIGPLAN Not. New York, NY, USA, 2013. Feb. Vol. 48, no. 8. P. 219–228. DOI: 10.1145/2517327.2442538.
9. Arora N.S., Blumofe R.D., Plaxton C.G. Thread scheduling for multiprogrammed multiprocessors // Proceedings of the Tenth Annual ACM Symposium on Parallel Algorithms and Architectures. Puerto Vallarta, Mexico: Association for Computing Machinery, 1998. P. 119–129. DOI: 10.1145/277651.277678.
10. Yang J., He Q. Scheduling Parallel Computations by Work Stealing: A Survey // International Journal of Parallel Programming. 2018. Apr. Vol. 46, no. 2. P. 173–197. DOI: 10.1007/s10766-016-0484-8.
11. Knuth D.E. The art of computer programming, volume 1 (3rd ed.): fundamental algorithms. USA: Addison Wesley Longman Publishing Co., Inc., 1997. DOI: 10.5555/260999.
12. Alam M., Varshney A.K. A New Approach of Dynamic Load Balancing Scheduling Algorithm for Homogeneous Multiprocessor System // International Journal of Applied Evolutionary Computation. 2016. Vol. 7, no. 2. P. 61–75. DOI: 10.4018/IJAEC.2016040104.
13. Амелина Н.О., Корнивец А.Д., Иванский Ю.В., Тюшев К.И. Применение консенсусного протокола для балансировки загрузки стохастической децентрализованной сети при передаче данных // XII Всероссийское совещание по проблемам управления: Труды научной конференции, Москва, 16–19 июня, 2014. Москва: ИПУ РАН, 2014. С. 8902–8911.
14. Amelina N., Fradkov A., Jiang Y., Vergados D.J. Approximate Consensus in Stochastic Networks with Application to Load Balancing // IEEE Transactions on Information Theory. 2015. Vol. 61, no. 4. P. 1739–1752. DOI: 10.1109/TIT.2015.2406323.
15. Li J., Agrawal K., Elnikety S., *et al.* Work stealing for interactive services to meet target latency // SIGPLAN Not. New York, NY, USA, 2016. Feb. Vol. 51, no. 8. Article 14. DOI: 10.1145/3016078.2851151.
16. Wimmer M., Versaci F., Traff J.L., *et al.* Data structures for task-based priority scheduling // SIGPLAN Not. New York, NY, USA, 2014. Feb. Vol. 49, no. 8. P. 379–380. DOI: 10.1145/2692916.2555278.
17. Gmys J., Leroy R., Mezmaz M., *et al.* Work stealing with private integer–vector–matrix data structure for multi-core branch-and-bound algorithms // Concurrency and Computation: Practice and Experience. 2016. Vol. 28, no. 18. P. 4463–4484. DOI: https://doi.org/10.1002/cpe.3771.
18. Mitzenmacher M. Analyses of load stealing models based on differential equations // Proceedings of the Tenth Annual ACM Symposium on Parallel Algorithms and Architectures. Puerto Vallarta, Mexico: Association for Computing Machinery, 1998. P. 212–221. DOI: 10.1145/277651.277687.
19. Kuchumov R., Sokolov A., Korkhov V. Staccato: Cache-Aware Work-Stealing Task Scheduler for Shared-Memory Systems // Computational Science and Its Applications – ICCSA 2018 / ed. by O. Gervasi, B. Murgante, S. Misra, *et al.* Cham: Springer International Publishing, 2018. P. 91–102. DOI: 10.1007/978-3-319-95171-3_8.
20. Kuchumov R., Sokolov A., Korkhov V. Staccato: shared-memory work-stealing task scheduler with cache-aware memory management // International Journal of Web and Grid Services. 2019. Vol. 15, no. 4. P. 394–407. DOI: 10.1504/IJWGS.2019.103233.

21. Аксенова Е.А., Соколов А.В. Методы управления work-stealing деками в динамических планировщиках многопроцессорных параллельных вычислений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 4. С. 76–93. DOI: 10.14529/cmse230403.

22. Chase D., Lev Y. Dynamic circular work-stealing deque // Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures. Las Vegas, Nevada, USA: Association for Computing Machinery, 2005. P. 21–28. DOI: 10.1145/1073970.1073974.

23. Sokolov A., Drac A. The linked list representation of n LIFO-stacks and/or FIFO-queues in the single-level memory // Information Processing Letters. 2013. Vol. 113, no. 19. P. 832–835. DOI: 10.1016/j.ipl.2013.07.021.

24. Hendler D., Lev Y., Moir M., Shavit N. A dynamic-sized nonblocking work stealing deque // Distributed Computing. 2006. Feb. Vol. 18, no. 3. P. 189–207. DOI: 10.1007/s00446-005-0144-5.

25. Аксенова Е.А., Лазутина А.А., Соколов А.В. Об оптимальных методах представления динамических структур данных // Обозрение прикладной и промышленной математики. 2003. Т. 10, № 2. С. 375–376.

26. Sokolov A., Barkovsky E. The Mathematical Model and the Problem of Optimal Partitioning of Shared Memory for Work-Stealing Deques // Parallel Computing Technologies / ed. by V. Malyshkin. Cham: Springer International Publishing, 2015. P. 102–106. DOI: 10.1007/978-3-319-21909-7_11.

27. Барковский Е.А., Кучумов Р.И., Соколов А.В. Оптимальное управление двумя work-stealing деками в общей памяти при различных стратегиях перехвата работы // Программные системы: теория и приложения. 2017. Т. 8, № 1. С. 83–103. DOI: 10.25209/2079-3316-2017-8-1-83-103.

28. Барковский Е.А., Лазутина А.А., Соколов А.В. Построение и анализ модели процесса работы с двумя деками, двигающимися друг за другом в общей памяти // Программные системы: теория и приложения. 2019. Т. 10, № 1. С. 3–17. DOI: 10.25209/2079-3316-2019-10-1-3-17.

29. Aksenova E.A., Barkovsky E.A., Sokolov A.V. The Models and Methods of Optimal Control of Three Work-Stealing Deques Located in a Shared Memory // Lobachevskii Journal of Mathematics. 2019. Nov. Vol. 40, no. 11. P. 1763–1770. DOI: 10.1134/S1995080219110052.

30. Лазутина А.А., Соколов А.В. Об оптимальном управлении Work-Stealing деками в двухуровневой памяти // Вестник компьютерных и информационных технологий. 2020. Т. 17, № 4. С. 51–60. DOI: 10.14489/vkit.2020.04.pp.051-060.

31. Аксенова Е.А., Лазутина А.А., Соколов А.В. Минимизация средних затрат на перераспределение при работе с work-stealing деком в двухуровневой памяти // Программные системы: теория и приложения. 2021. Т. 12, № 2. С. 53–71. DOI: 10.25209/2079-3316-2021-12-2-53-71.

32. Aksenova E.A., Lazutina A.A., Sokolov A.V. About Optimal Management of Work-Stealing Deques in Two-Level Memory // Lobachevskii Journal of Mathematics. 2021. July. Vol. 42, no. 7. P. 1475–1482. DOI: 10.1134/S1995080221070027.

Аксёнова Елена Алексеевна, к.ф.-м.н., лаборатория информационных компьютерных технологий, Институт прикладных математических исследований Карельского научного центра Российской академии наук (Петрозаводск, Российская Федерация)

Барковский Евгений Александрович, независимый исследователь (Петрозаводск, Российская Федерация)

Соколов Андрей Владимирович, д.ф.-м.н., профессор, лаборатория информационных компьютерных технологий, Институт прикладных математических исследований Карельского научного центра Российской академии наук (Петрозаводск, Российская Федерация)

DOI: 10.14529/cmse240303

OPTIMAL CONTROL OF THREE WORK-STEALING DEQUES LOCATED IN TWO-LEVEL MEMORY

© 2024 E.A. Aksenova, E.A. Barkovsky, A.V. Sokolov

Institute of Applied Mathematical Research
of the Karelian Research Centre of the Russian Academy of Sciences
(str. Pushkinskaya 11, Petrozavodsk, 185910 Russia)
E-mail: aksenova@krc.karelia.ru, barkevgen@gmail.com, avs@krc.karelia.ru

Received: 26.07.2024

The problem of even balancing of tasks between threads may arise during parallel computations. One way to resolve this issue is to implement distributed dynamic load balancing. In this balancing method each worker thread has its own task queue, and threads themselves distribute further tasks. One of the widely used distributed dynamic balancing methods is "work-stealing": a thread that runs out of tasks begins to "steal" them from another thread. To utilize this method, each thread must have its own specialized deque where pointers to tasks are stored. In this paper, we propose and examine a new method for representing three work stealing deques in two-level memory. We consider the case of working with three deques, where one section of fast memory contains two LIFO parts of the deques, i.e., two stacks that grow towards each other. Third LIFO part and three FIFO parts are located in the other section of fast memory. FIFO parts are combined into one FIFO queue, from which elements are only deleted (being stolen). The middle parts of the deques are located in slow memory. They are accessed when LIFO or FIFO parts located in fast memory are empty or overflowed. We consider the problem of finding optimal partition of fast memory for three deques with predetermined probabilities of operations. This choice depends on the characteristics of memory levels, operation probabilities, and optimality criteria. The optimality criterion is the maximum mean system operating time (the average number of operations) before memory reallocation.

Keywords: data structures, Monte Carlo methods, two-level memory, work stealing deques.

FOR CITATION

Aksenova E.A., Barkovsky E.A., Sokolov A.V. Optimal Control of Three Work-Stealing Deques Located in Two-Level Memory. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 3. P. 47–60. (in Russian) DOI: 10.14529/cmse240303.

References

1. Herlihy M., Shavit N. The Art of Multiprocessor Programming. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008. DOI: 10.5555/1734069.
2. Kwok Y.-K., Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surv. New York, NY, USA, 1999. Dec. Vol. 31, no. 4. P. 406–471. DOI: 10.1145/344588.344618.
3. Alakeel A.M. A Guide to Dynamic Load Balancing in Distributed Computer Systems. International Journal of Computer Science and Network Security. 2010. Vol. 10, no. 6. P. 153–160.
4. Beaumont O., Carter L., Ferrante J., *et al.* Centralized versus distributed schedulers for multiple bag-of-task applications. Proceedings 20th IEEE International Parallel & Distributed Processing Symposium. 2006. P. 10. DOI: 10.1109/IPDPS.2006.1639262.
5. Xia Y., Prasanna V.K., Li J. Hierarchical Scheduling of DAG Structured Computations on Manycore Processors with Dynamic Thread Grouping. Job Scheduling Strategies for Parallel Processing / ed. by E. Frachtenberg, U. Schwiegelshohn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 154–174. DOI: 10.1007/978-3-642-16505-4_9.
6. Hendler D., Shavit N. Non-blocking steal-half work queues. Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing. Monterey, California: Association for Computing Machinery, 2002. P. 280–289. DOI: 10.1145/571825.571876.
7. Hendler D., Shavit N. Work dealing. Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures. Winnipeg, Manitoba, Canada: Association for Computing Machinery, 2002. P. 164–172. DOI: 10.1145/564870.564900.
8. Acar U.A., Charguéraud A., Rainey M. Scheduling parallel programs by work stealing with private deques. SIGPLAN Not. New York, NY, USA, 2013. Feb. Vol. 48, no. 8. P. 219–228. DOI: 10.1145/2517327.2442538.
9. Arora N.S., Blumofe R.D., Plaxton C.G. Thread scheduling for multiprogrammed multiprocessors. Proceedings of the Tenth Annual ACM Symposium on Parallel Algorithms and Architectures. Puerto Vallarta, Mexico: Association for Computing Machinery, 1998. P. 119–129. DOI: 10.1145/277651.277678.
10. Yang J., He Q. Scheduling Parallel Computations by Work Stealing: A Survey. International Journal of Parallel Programming. 2018. Apr. Vol. 46, no. 2. P. 173–197. DOI: 10.1007/s10766-016-0484-8.
11. Knuth D.E. The art of computer programming, volume 1 (3rd ed.): fundamental algorithms. USA: Addison Wesley Longman Publishing Co., Inc., 1997. DOI: 10.5555/260999.
12. Alam M., Varshney A.K. A New Approach of Dynamic Load Balancing Scheduling Algorithm for Homogeneous Multiprocessor System. International Journal of Applied Evolutionary Computation. 2016. Vol. 7, no. 2. P. 61–75. DOI: 10.4018/IJAEC.2016040104.
13. Amelina N.O., Kornivec A.D., Ivanskij J.V., Tjushev K.I. Primenenie konsensusnogo protokola dlja balansirovki zagruzki stohasticheskoj decentralizovannoj seti pri peredache dannyh. XII Vserossijskoe soveshhanie po problemam upravlenija: Scientific conference proceedings, Moscow, 16–19 June, 2014. Moscow: ICS of RAS, 2014. P. 8902–8911. (in Russian).

14. Amelina N., Fradkov A., Jiang Y., Vergados D.J. Approximate Consensus in Stochastic Networks with Application to Load Balancing. IEEE Transactions on Information Theory. 2015. Vol. 61, no. 4. P. 1739–1752. DOI: 10.1109/TIT.2015.2406323.

15. Li J., Agrawal K., Elnikety S., *et al.* Work stealing for interactive services to meet target latency. SIGPLAN Not. New York, NY, USA, 2016. Feb. Vol. 51, no. 8. Article 14. DOI: 10.1145/3016078.2851151.

16. Wimmer M., Versaci F., Traff J.L., *et al.* Data structures for task-based priority scheduling. SIGPLAN Not. New York, NY, USA, 2014. Feb. Vol. 49, no. 8. P. 379–380. DOI: 10.1145/2692916.2555278.

17. Gmys J., Leroy R., Mezmaz M., *et al.* Work stealing with private integer–vector–matrix data structure for multi-core branch-and-bound algorithms. Concurrency and Computation: Practice and Experience. 2016. Vol. 28, no. 18. P. 4463–4484. DOI: https://doi.org/10.1002/cpe.3771.

18. Mitzenmacher M. Analyses of load stealing models based on differential equations. Proceedings of the Tenth Annual ACM Symposium on Parallel Algorithms and Architectures. Puerto Vallarta, Mexico: Association for Computing Machinery, 1998. P. 212–221. DOI: 10.1145/277651.277687.

19. Kuchumov R., Sokolov A., Korkhov V. Staccato: Cache-Aware Work-Stealing Task Scheduler for Shared-Memory Systems. Computational Science and Its Applications – ICCSA 2018 / ed. by O. Gervasi, B. Murgante, S. Misra, *et al.* Cham: Springer International Publishing, 2018. P. 91–102. DOI: 10.1007/978-3-319-95171-3_8.

20. Kuchumov R., Sokolov A., Korkhov V. Staccato: shared-memory work-stealing task scheduler with cache-aware memory management. International Journal of Web and Grid Services. 2019. Vol. 15, no. 4. P. 394–407. DOI: 10.1504/IJWGS.2019.103233.

21. Aksenova E.A., Sokolov A.V. Control Methods of Work-stealing Deques in Dynamic Schedulers of Multiprocessor Parallel Computations. Bulletin of the SUSU. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 4. P. 76–93. (in Russian) DOI: 10.14529/cmse230403.

22. Chase D., Lev Y. Dynamic circular work-stealing deque. Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures. Las Vegas, Nevada, USA: Association for Computing Machinery, 2005. P. 21–28. DOI: 10.1145/1073970.1073974.

23. Sokolov A., Drac A. The linked list representation of n LIFO-stacks and/or FIFO-queues in the single-level memory. Information Processing Letters. 2013. Vol. 113, no. 19. P. 832–835. DOI: 10.1016/j.ipl.2013.07.021.

24. Hendler D., Lev Y., Moir M., Shavit N. A dynamic-sized nonblocking work stealing deque. Distributed Computing. 2006. Feb. Vol. 18, no. 3. P. 189–207. DOI: 10.1007/s00446-005-0144-5.

25. Aksenova E.A., Lazutina A.A., Sokolov A.V. Ob optimal'nyh metodah predstavlenija dinamicheskih struktur dannyh. Obozrenie prikladnoj i promyshlennoj matematiki. 2003. Vol. 10, no. 2. P. 375–376. (in Russian).

26. Sokolov A., Barkovsky E. The Mathematical Model and the Problem of Optimal Partitioning of Shared Memory for Work-Stealing Deques. Parallel Computing Technologies / ed. by V. Malyshkin. Cham: Springer International Publishing, 2015. P. 102–106. DOI: 10.1007/978-3-319-21909-7_11.

27. Barkovksy E., Kuchumov R., Sokolov A. Optimal Control of Two Deques in Shared Memory with Various Work-Stealing Strategies. Program Systems: Theory and Applications. 2017. Vol. 8, no. 1. P. 83–103. (in Russian) DOI: 10.25209/2079-3316-2017-8-1-83-103.

28. Barkovsky E., Lazutina A., Sokolov A. The Optimal Control of Two Work-Stealing Deques, Moving One After Another in a Shared Memory. Program Systems: Theory and Applications. 2019. Vol. 10, no. 1. P. 19–32. DOI: 10.25209/2079-3316-2019-10-1-19-32.

29. Aksenova E.A., Barkovsky E.A., Sokolov A.V. The Models and Methods of Optimal Control of Three Work-Stealing Deques Located in a Shared Memory. Lobachevskii Journal of Mathematics. 2019. Nov. Vol. 40, no. 11. P. 1763–1770. DOI: 10.1134/S1995080219110052.

30. Lazutina A.A., Sokolov A.V. About Optimal Management of Work-Stealing Deques in Two-Level Memory. Herald of Computer and Information Technologies. 2020. Vol. 17, no. 4. P. 51–60. (in Russian) DOI: 10.14489/vkit.2020.04.pp.051-060.

31. Aksenova E.A., Lazutina A.A., Sokolov A.V. About Optimal Managment of Work-Stealing Deques in Two-Level Memory. Program Systems: Theory and Applications. 2021. Vol. 12, no. 2. P. 53–71. (in Russian) DOI: 10.25209/2079-3316-2021-12-2-53-71.

32. Aksenova E.A., Lazutina A.A., Sokolov A.V. About Optimal Management of Work-Stealing Deques in Two-Level Memory. Lobachevskii Journal of Mathematics. 2021. July. Vol. 42, no. 7. P. 1475–1482. DOI: 10.1134/S1995080221070027.