УДК 004.258, 004.451.7 DOI: 10.14529/cmse230403

МЕТОДЫ УПРАВЛЕНИЯ WORK-STEALING ДЕКАМИ В ДИНАМИЧЕСКИХ ПЛАНИРОВЩИКАХ МНОГОПРОЦЕССОРНЫХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

© 2023 Е.А. Аксёнова, А.В. Соколов

Институт прикладных математических исследований,
обособленное подразделение исследовательского центра
«Карельский научный центр Российской академии наук»
(185910 Петрозаводск, ул. Пушкинская, д. 11)
E-mail: aksenova@krc.karelia.ru, sokavs@gmail.com
Поступила в редакцию: 21.07.2023

В параллельных планировщиках задач, работающих по стратегии work-stealing, каждый процессор имеет свой дек задач. Один конец дека используется для добавления и извлечения задач только владельцем, а другой — для перехвата задач другими процессорами. В статье предлагается обзор методов управления work-stealing деками, которые используются при реализации work-stealing планировщиков параллельных задач, а также представлено описание поставленных и решенных нашим коллективом задач оптимального управления деками для стратегии work-stealing. Принцип алгоритмов оптимального управления деками в двухуровневой памяти заключается в том, что при переполнении выделенного участка быстрой памяти происходит перераспределение элементов (задач) дека между уровнями памяти. В быстрой памяти остаются элементы из концов дека, так как с ними будет происходить работа в ближайшее время, а элементы средней части дека хранятся в медленной памяти. В таком случае необходимо определить оптимальное количество элементов, которое нужно оставить в быстрой памяти, в зависимости от критерия оптимальности и параметров системы.

Ключевые слова: имитационные и марковские модели оптимального управления структурами данных, оптимальное кэширование деков, оптимальное управление work-stealing деками, оптимизация work-stealing планировщиков, управляемые случайные блуждания.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Аксёнова 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.

Введение

Дек (от англ. deque — double ended queue) — это структура данных, в которой добавление новых элементов и удаление существующих производится с обоих концов (рис. 1). Дек поддерживает FIFO- и LIFO-операции, поэтому с помощью дека можно реализовать как стек, так и очередь.

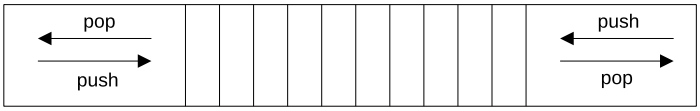

Рис. 1. Дек

В параллельных планировщиках задач, работающих по стратегии work-stealing, каждый процессор имеет свой дек (deque) задач. Один конец дека используется для добавления и извлечения задач только владельцем, а другой — для перехвата задач другими процессорами. Когда процессор создает новую задачу, он добавляет задачу в свой дек; когда процессору нужна задача, он берет задачу из дека. Если процессор узнает, что его дек пуст, он перехватывает задачи у другого процессора. Добавление и удаление элементов (задач или указателей на задачи) выполняется на одном конце дека, который работает как LIFO-стек, а перехват (кража) элементов происходит на другом конце дека — как в FIFO-очереди (рис. 2).

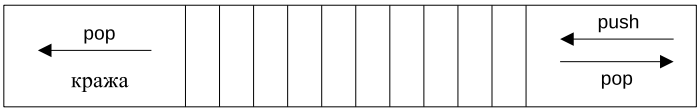

Рис. 2. Work-stealing дек

В данной статье предлагается обзор методов управления work-stealing деками, которые используются при реализации work-stealing планировщиков параллельных задач, а также представлено описание поставленных и решенных нашим коллективом задач оптимального управления work-stealing деками. Принцип алгоритмов оптимального управления деками в двухуровневой памяти заключается в том, что при переполнении отведенного деку участка быстрой памяти происходит перераспределение элементов между уровнями памяти: в быстрой памяти остаются элементы из концов дека, так как с ними будет происходить работа в ближайшее время, а элементы средней части дека хранятся в памяти второго уровня (рис. 3).

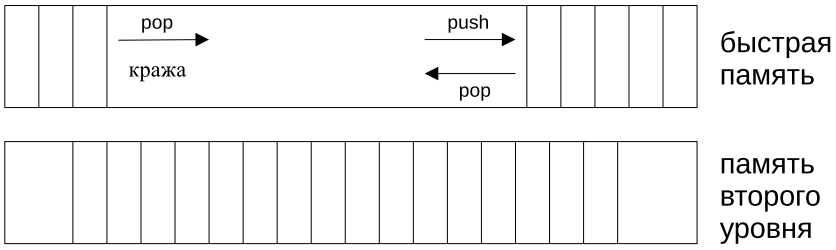

Рис. 3. Work-stealing дек в двухуровневой памяти

Основной целью наших работ по этой тематике являлось построение и анализ математических и имитационных моделей с целью оптимизации работы с деками, расположенными в общей многоуровневой памяти. Параметрами для этих моделей являются число деков, вероятности операций с деками на каждом шаге дискретного времени (возможно как последовательное,так и параллельное выполнение операций), размеры памяти всех уровней, временные характеристики операций работы с деками в разных уровнях памяти. В качестве критериев оптимальности рассматривались максимизация суммы средних времен работы каждого дека до перераспределения памяти, максимизация наименьшего среднего времени работы каждого дека до перераспределения памяти, минимизация суммы средних затрат на перераспределения памяти, возникающие в случае переполнения или опустошения быстрой памяти каждым деком.

Традиционно деки разрабатываются с предположением, что указатели задач хранятся в этих структурах данных, а объекты задач находятся в куче памяти. Путем изменения организации деков задач так, чтобы они могли содержать объекты задач вместо указателей, удалось повысить производительность более чем в 2.5 раза для приложений, привязанных к ЦП, и уменьшить количество промахов кэша последнего уровня на 30% по сравнению с work-stealing планировщиками Intel Threading Building Blocks и Intel/MIT Cilk.

Статья организована следующим образом. В разделе 1 дан обзор статей, в которых описаны различные методы балансировки задач в многопроцессорных системах. В разделе 2 представлены задачи оптимального управления деками для стратегии work-stealing, решенные нашим коллективом, а так же описание реализации экспериментального динамического work-stealing планировщика. В заключении приводится краткая сводка результатов, полученных в работе, и указаны направления дальнейших исследований.

1. Методы балансировки задач в многопроцессорных системах

Стратегии балансировки параллельных вычислений разделяют на статические и динамические [1]. Статическая балансировка используется, когда известны все свойства и особенности выполняемых задач. В этом случае можно заранее определить оптимальное расписание задач (например, минимизировать среднее время решения). Такие задачи считаются NP-полными [2] и встречаются достаточно редко. При динамической балансировке планировщик во время работы использует некую относительно простую стратегию балансировки, которая дает результат близкий к оптимальному (например, наименьшее время выполнения задачи) [3]. В свою очередь, в динамической балансировке выделяют централизованную, когда есть специальный поток для распределения работы между другими потоками, и нецентрализованную [4, 5]. При нецентрализованной балансировке у каждого потока имеется своя очередь задач (подпрограмм) для выполнения и потоки сами осуществляют распределение. Например, поток, у которого слишком много задач, может их перераспределить между другими потоками (метод «work-dealing» [6, 7]). Если у потока заканчиваются задачи для выполнения, он их запрашивает (метод «work-requesting» [8]) или перехватывает (метод «work-stealing» [9]) у другого потока.

Теоретически доказано [9], что стратегия work-stealing дает распределение задач близкое к оптимальному (минимизация времени выполнения задачи), и на практике она себя также зарекомендовала [10]. Ее реализацию можно встретить во многих планировщиках задач, например, Cilk [11], dotNET TPL [12], X10 [13] и других. В этом методе балансировки нагрузки каждый процессор решает ряд задач, указатели на которые хранятся в деке этого процессора. Когда процессор создает новую задачу, он добавляет указатель на задачу в свой дек; когда процессору нужна задача, он берет указатель на задачу из вершины дека. Если процессор узнает, что его дек пуст, он перехватывает указатели на задачи у другого процессора. Первые две операции выполняются как в LIFO-стеке, а перехват происходит из основания дека — как в FIFO-очереди. В терминологии Д. Кнута такая структура данных называется деком с ограниченным входом [14]. Количество элементов, извлекаемых за один перехват, может различаться. Так, в [15] предлагалось перехватывать один элемент, в [6] — половину элементов.

Механизм балансировки загрузки применяется в многопроцессорных системах и вычислительных сетях. Для повышения эффективности «work-stealing» метода балансировки

загрузки исследуют способы реализации и управления структурами данных. На основе этого метода разрабатываются различные планировщики задач. Например, в [15] исследуется проблема эффективного планирования структурированных многопоточных вычислений на параллельных компьютерах, предлагается планировщик с перехватом работы для многопоточных вычислений с зависимостями.

В работе [16] авторы разделяют методы балансировки нагрузки параллельных задач на статическую и динамическую, а также предлагают стратегию динамической балансировки нагрузки для однородной многопроцессорной системы и применяют ее к сети, называемой сетью Folded Crossed Cube. Результаты экспериментов показывают, что вместе со временем выполнения достигается меньший коэффициент дисбаланса нагрузки. В статье [17] рассмотрено применение теоретических результатов к задаче балансировки нагрузки в стохастических динамических сетях с неполной информацией о текущих состояниях агентов и изменяющимся набором каналов связи. Установлены условия достижения оптимального уровня балансировки нагрузки. Производительность системы оценивается как аналитически, так и путем моделирования.

В статье [18] исследуются интерактивные веб-сервисы, которые все чаще управляют критически важными бизнес-задачами, такими как поиск, реклама, игры, покупки и финансы. Показано, как обобщить кражу работы, которая используется для минимизации времени выполнения одного параллельного задания, для оптимизации целевой задержки в интерактивных сервисах с несколькими параллельными запросами. Авторы разрабатывают новый адаптивный метод кражи задач — метод контроля хвостов, который уменьшает количество запросов, не достигающих целевой задержки. Этот подход реализуется в библиотеке Intel Threading Building Blocks и оценивается на различных рабочих нагрузках. Метод контроля хвостов существенно снижает количество запросов, превышающих желаемую целевую задержку, и обеспечивает относительное улучшение до 58% по сравнению с различными базовыми методами.

В работе [19] анализируется Intel Threading Building Blocks — библиотека C++ для параллельного программирования. Шаблоны библиотеки для общих параллельных циклов построены на вложенном параллелизме и планировщике, перехватывающем задачи. В статье обсуждаются способы оптимизации, в которых алгоритм для оптимизации работы учитывает данные о кражах задач.

В статье [20] авторы утверждают, что кража задач — это эффективный метод реализации балансировки нагрузки при детальном параллелизме задач. Обычно для этой цели используются параллельные деки. Недостатком многих параллельных деков является то, что они требуют дорогостоящих ограничений памяти для локальных операций с деками. В этой статье предлагается новый неблокирующий дек для кражи задач, основанный на разделенной очереди задач. В деке используется динамическая точка разделения между общей и частной частями дека. Авторы представляют «Lace» — реализацию кражи задач с интерфейсом, похожим на work-stealing библиотеку кражи задач Wool [21].

В статье [22] основное внимание уделяется многоядерным алгоритмам ветвей и границ для решения крупномасштабных задач оптимизации на основе перестановок. Исследуется пять стратегий кражи задач с новой структурой данных, которая называется целочисленно-векторной матрицей. В этих стратегиях каждый поток имеет частную матрицу, позволяющую локально управлять набором подзадач. Стратегии различаются способом выбора потока-жертвы и степенью детализации украденных задач. Результаты оценки эффектив-

ности подхода на основе целочисленно-векторной матрицы показали, что предложенный алгоритм кражи задач превосходит алгоритм кражи на основе связанных списков по процессорному времени, использованию памяти и количеству выполняемых операций кражи.

В работе [23] представлены три структуры данных без блокировок для распределения приоритетных задач: приоритетная структура с кражей задач, централизованная с расслабленной семантикой и гибридная, сочетающая обе концепции. На примере задачи поиска в графе кратчайшего пути от одной вершины (SSSP, Single Source Shortest Path) показано, как различные подходы влияют на расстановку приоритетов, и предоставлены верхние границы количества проверенных узлов. Авторы утверждают, что распределение приоритетных задач обеспечивает интуитивно понятный и простой способ распараллеливания проблемы SSSP, которая является сложной задачей. Экспериментальные данные подтверждают хорошую масштабируемость полученного алгоритма.

В работе [24] предлагается планировщик (LAWS, Locality-Aware Work-Stealing), который использует общую кэш-память и систему памяти NUMA. В LAWS распределитель задач с балансировкой нагрузки используется для равномерного разделения и хранения набора данных программы на всех узлах памяти и выделения задачи интерфейсу, в котором узел локальной памяти хранит свои данные. Для балансировки нагрузки применяется трехуровневый планировщик с кражей задач. Результаты экспериментов показывают, что LAWS может повысить производительность программ, связанных с памятью, до 54.2% по сравнению с традиционными work-stealing планировщиками.

Асимметричные многоядерные архитектуры (AMC, Asymmetric Multi-Core), в которых ядра разных процессоров имеют разную производительность и энергопотребление, широко используются от крупных центров обработки данных до мобильных смартфонов из-за их высокой производительности и энергоэффективности. Однако существующие способы распределения задач часто приводят к низкой производительности параллельных программ на новых архитектурах AMC из-за несбалансированной рабочей нагрузки, кэш-промахов и удаленного доступа к памяти. Чтобы решить эту проблему, авторы [25] предлагают систему выполнения (SAWS, Selective Asymmetry-aware Work-Stealing), которая может сократить удаленный доступ к памяти, одновременно распределяя рабочую нагрузку между асимметричными ядрами. SAWS состоит из планировщика задач с учетом асимметричности и планировщика выбора кражи задач. Планировщик задач с учетом асимметричности правильно распределяет задачи по асимметричным процессорам, чтобы большинство задач могли получить доступ к данным из узла локальной памяти, а рабочая нагрузка балансировалась в соответствии с вычислительными возможностями различных процессоров. После этого используется планировщик выбора кражи задач для дальнейшей балансировки рабочей нагрузки во время выполнения. Результаты экспериментов на реальной системе показывают, что SAWS повышает производительность программ, связанных с памятью, до 59.3% по сравнению с традиционными work-stealing планировщиками в архитектурах AMC.

В статье [26] представлен масштабируемый планировщик с адаптивным алгоритмом кражи задач (SLAW, Scalable Locality-aware Adaptive Work-stealing scheduler), который выбирает способ распределения каждой задачи во время выполнения. Планировщик SLAW также устанавливает ограничения на использование стека и кучи, необходимые для хранения задач.

В работе [27] разработаны модели и проведен анализ нескольких рандомизированных алгоритмов кражи работы в динамических условиях. С помощью дифференциальных

уравнений в этих моделях показано предельное поведение систем при возрастании числа процессоров до бесконечности. Этот подход дает возможность моделировать разные системы и обеспечивать точные численные аппроксимации поведения системы даже тогда, когда количество процессоров относительно невелико.

В [28] предлагается новый механизм оптимизации планировщиков путем сбора информации во время выполнения программы. С этой целью, используется матрица взаимодействия потоков (TIM, Thread-Interaction Matrix), в которой хранится статистика взаимодействия потоков.

2. Модели оптимального управления work-stealing деками

В [29] описана реализация экспериментального динамического work-stealing планировщика. Параллельные планировщики задач, работающие по стратегии work-stealing, дают распределение задач близкое к оптимальному, и при этом обладают низкими накладными расходами по времени, используемой памяти и межпроцессорным синхронизациям. Основная идея этой стратегии: когда у процессора заканчиваются задачи для выполнения, он их забирает (крадет) у другого процессора. Один из недостатков этой стратегии — большое количество краж, возникающих при выполнении относительно маленьких задач. В этой статье описана реализация work-stealing планировщика на языке C++, в которой возможно извлекать определенное количество задач за одну кражу, и предложен способ оценки вероятностей операций, выполняющихся в планировщике. Такая модификация при правильном выборе количества украденных задач позволяет значительно уменьшить общее количество краж.

Для представления динамических деков в памяти используются разные способы. Можно воспользоваться методом «связное представление» [30]. Модель такого метода будет схожа с уже построенной моделью связного представления стеков и очередей [31]. Можно использовать способ двухсвязного страничного представления [32]. Математическая модель для такого представления деков будет близка к разработанной модели для других структур данных [27], только для стеков и очередей достаточно иметь односвязный список страниц. Также, возможно представление параллельных структур в общей памяти (которая заранее не делится), когда они двигаются друг за другом по кругу, начиная с некоторого начального места в памяти. Этот метод работы был предложен в [33] и анализировался в [34, 35] для FIFO-очередей. Использование этого метода для work-stealing деков было запатентовано [36].

Отметим, что среди многоядерных архитектур есть архитектуры без кэш-памяти. Например, в архитектуре AsAP-II каждое ядро имеет два FIFO-буфера, а в архитектуре SEAforth — два стека (для хранения данных и адресов возврата). В этих архитектурах очереди и стеки реализованы циклически и отделены друг от друга с возможностью потери элементов из-за переполнения. Но в наших работах исследуются ситуации, когда для хранения нескольких структур данных используется разделяемая память. В некоторых случаях это может свести к минимуму количество потерянных элементов. На основе этих архитектур аппаратно могут быть реализованы work-stealing деки. В этом случае важно исследовать оптимальную организацию двух деков, а затем, в случае произвольного количества ядер, можно получить желаемые чипы, составив их из «двухдековых». В [37] рассмотрена задача и предложена математическая модель оптимального разбиения памяти одного уровня для двух work-stealing деков.

В [38] представлен анализ математических моделей процесса работы с двумя циклическими деками, расположенными в общей памяти. Параметрами этих моделей являются вероятности операций на каждом шаге дискретного времени (возможно как последовательное, так и параллельное выполнение операций). Модели строятся в виде случайных блужданий по целочисленной решетке на плоскости. На основе вышеупомянутых моделей решены задачи оптимального разделения памяти при некоторых стратегиях перехвата элементов. В качестве критерия оптимальности рассматривается максимальное среднее время до переполнения памяти. Проведены статистические исследования по оценке вероятностей операций работы с деками для нескольких типов задач, выполняемых в реализованном планировщике — вычисление чисел Фибоначчи с помощью рекурсии, задача о рюкзаке, решаемая методом ветвей и границ, умножение матриц, сортировка слиянием, обход графа задач. Для полученных вероятностей операций работы с деками проведены численные эксперименты по анализу разработанных моделей. Разработаны алгоритмы и программы на языке C для построения матриц переходных вероятностей P при произвольных значениях m (размер быстрой памяти), s (размер первого дека), o (количество перехваченных элементов), вероятностей операций, а также нахождения оптимального разбиения памяти между деками, в зависимости от вероятностных характеристик деков, и оптимального количества элементов для перехвата. Для решения поставленных задач использовался аппарат управляемых случайных блужданий, поглощающих цепей Маркова, система LAPACK.

В [39] предложены математические модели работы с n последовательными циклическими деками, расположенными в общей памяти. Математические модели строятся в виде случайных блужданий по целочисленной решетке в n-мерном пространстве. Операции с заданными вероятностями происходят на каждом шаге дискретного времени. Решалась задача нахождения оптимального разбиения памяти между n деками и задача определения оптимального количества элементов для кражи. Критерием оптимальности являлось максимальное среднее время до переполнения памяти.

В [40] анализировались два метода представления деков: один из распространенных методов — раздельное последовательное циклическое представление деков, и новый 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. В работе представлен анализ модели процесса работы с двумя последовательными деками, когда они двигаются друг за другом по кругу в общей памяти. Предложены математическая и имитационная модели данного процесса и проведены численные эксперименты. Математическая модель была построена, как случайное блуждание по целым точкам в пирамиде. Имитационная модель строится с помощью метода Монте-Карло. Используемая стратегия work-stealing — перехват одного элемента. В качестве критерия оптимальности было рассмотрено максимальное среднее время работы до переполнения памяти.

В работе [41] представлены разработка, анализ и сравнение моделей и методов управления деками в ограниченной разделяемой памяти. Для решения поставленных задач использовалась модель в виде управляемого случайного блуждания и имитационное моделирование. Для случая трех деков рассмотрены следующие способы управления:

– каждый из трех деков располагается в своей отдельной области памяти;
– три дека двигаются друг за другом по кругу;

– комбинированный способ — два дека располагаются друг за другом, один отдельно.

В [42] исследовался способ способы повышения производительности work-stealing планировщиков за счет усовершенствования внутренних механизмов обработки данных. Традиционно деки разрабатываются с предположением, что указатели задач хранятся в этих структурах данных, а объекты задач находятся в куче памяти. Путем изменения организации деков задач таким образом, чтобы они могли содержать объекты задач вместо указателей, нам удалось повысить производительность более чем в 2.5 раза для приложений, привязанных к ЦП, и уменьшить количество промахов кэша последнего уровня на 30% по сравнению с work-stealing планировщиками Intel Threading Building Blocks и Intel/MIT Cilk. Сравнения производились на Intel Pentium N3530 2.16GHz, 4 CPU, 64 L1, с Linux 4.4.0 и на ARMv7l rev5, 4 CPU, 32 L1, с Linux 4.4.8. Использовались следующие тесты: умножение матриц, сортировка слиянием, рекурсивное вычисление чисел Фибоначчи, решение задачи о рюкзаке, обход графа в глубину. Реализации планировщика для общей и распределенной памяти размещены в свободных репозиториях [43] и [44] соответственно.

В работе [45] предлагается использовать разработанный нашим коллективом work-stealing планировщик [29, 42] для обучения сверточных нейронных сетей. В работе [46] авторы предлагают новые решения для оптимизации сельского хозяйства, основанные на использовании встроенных компьютерных систем и сенсорных сетей. Использование таких систем обработки информации поможет вытеснить мир, основанный на воздушных (беспилотниках) и наземных роботах. Эти инструменты будут выполнять точные задачи с помощью систем определения местоположения, которые предоставляют точную информацию. Чтобы всего этого добиться, необходимо провести тщательное изучение информационных систем сельскохозяйственных полей с использованием точных датчиков. Также нужны новые инструменты для управления данными, чтобы обрабатывать их в режиме реального времени. Правильное распределение этих данных с помощью различных инструментов (и в нашей работе [42] по мнению авторов статьи приводится один такой инструмент), требует их точной предварительной обработки, что позволит правильно обрабатывать информацию для повышения урожайности и качества продукции сельскохозяйственных полей. Также оптимизация работы с данными в реальном времени важна во многих других приложениях, например, в военных.

В [47] рассмотрена задача оптимального управления work-stealing деком в двухуровневой памяти. Предполагается, что известны вероятности параллельных операций с деком. Задача состоит в нахождении числа элементов FIFO- и LIFO-частей дека, которые при перераспределении дека будут оставлены в быстрой памяти, чтобы максимизировать среднее время работы до перераспределения памяти.

В [48] рассмотрена задача оптимального управления work-stealing деком в двухуровневой памяти. Предполагается, что известны вероятности параллельных операций с деком и временные характеристики уровней памяти. Задача состоит в нахождении оптимального количества элементов FIFO- и LIFO-частей 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.

В [49] анализируется задача оптимального управления двумя деками в двухуровневой памяти (например, регистры — оперативная память, когда известны вероятности параллельных операций с деками. Рассмотрен классический последовательный циклический метод представления дека в памяти. В случае переполнения дека в быстрой памяти или опустошения FIFO-части или LIFO-части происходят необходимые обмены между быстрой и медленной памятью и сдвиги элементов в быстрой памяти для перемещения в оптимальное состояние, которое необходимо найти. Задача состоит в том, чтобы найти оптимальное разбиение общей быстрой памяти для деков и определить оптимальное состояние каждого дека в каждом разделе после перераспределения памяти, т.е. найти оптимальное количество элементов, взятых с обеих сторон дека, которое надо оставлять в быстрой памяти, если дек заполнен или опустошен. Критерием оптимальности для совместного использования памяти является максимизация суммы средних времен работы каждого дека до перераспределения памяти и максимизация наименьшего среднего времени работы каждого дека до перераспределения памяти.

В [50] рассматривается задача оптимального управления двумя деками в двухуровневой памяти. Вероятности параллельных операций с деками известны. Задача состоит в том, чтобы найти оптимальное разделение быстрой памяти для деков и определить оптимальное количество элементов для обоих концов каждого дека, которое сохраняется в быстрой памяти после перераспределения памяти. В качестве критерия оптимальности рассматривается минимальная сумма средних затрат на перераспределения памяти, возникающие в случае переполнения или опустошения быстрой памяти для каждого дека. Этот критерий позволяет учитывать конкретные скорости доступа к уровням памяти и применять разработанные методы к различным сочетаниям быстрой и медленной памяти.

Заключение

В статье дан обзор моделей и методов оптимального параллельного управления work-stealing деками. Эта тематика является важной для оптимизации динамической балансировки параллельных вычислений. Например, возможен и такой вариант задач, когда некоторые деки могут быть очень большими и переполняют быструю память. Тогда декам нужно выделить отдельный раздел быстрой памяти и работать с ними так, как предложено для случая двух деков. Но в тоже время, некоторые деки имеют небольшой размер, поэтому их можно располагать друг за другом, пока они не переполнят быструю память.

В задаче параллельного управления двумя деками в двухуровневой памяти можно рассмотреть способ работы, когда в одном разделе быстрой памяти остаются LIFO-части деков — два стека, в которых происходят включения и исключения элементов, а в другом разделе быстрой памяти остаются две FIFO-части, из которых элементы только исключаются (кражи). Средние части деков находятся в медленной памяти, обращение к ним происходит при переполнении или опустошении LIFO-частей или FIFO-частей деков, расположенных в быстрой памяти. Возможны разные способы представления LIFO- и FIFO-частей в памяти: две FIFO-части расположены отдельно или объединены в одну очередь, две LIFO-части растут навстречу друг другу. Нужно определить, как разделить быструю память между двумя деками, какой способ организации LIFO- и FIFO-частей деков и какое начальное распределение памяти выбрать. Этот выбор зависит от характеристик уровней памяти, вероятностей операций со структурами данных и рассматриваемого критерия оптимальности. Можно рассмотреть три критерия оптимальности: максимизация среднего времени до

перераспределения быстрой памяти для двух деков; минимизация суммы средних затрат на перераспределение быстрой памяти для двух деков; минимизация наибольших средних затрат на перераспределение быстрой памяти для двух деков. Также можно попытаться обобщить эту задачу на произвольное число деков. В этом случае надо будет рассматривать разные варианты совместного расположения в быстрой памяти FIFO- и LIFO-частей деков.

Исследования, выполненные нашим коллективом, поддержаны грантами РФФИ № 15-01-03404 «Математические модели и оптимальные алгоритмы управления для некоторых методов работы с памятью в параллельных и сетевых устройствах» (2015–2017 гг.) и № 18-01-00125 «Математические модели и алгоритмы оптимального параллельного управления динамическими структурами данных и их реализация в планировщике многопроцессорных параллельных вычислений» (2018–2020 гг.).

Литература

1. Herlihy M., Shavit N. The Art of Multiprocessor Programming. Elsevier, 2008. 508 p.

2. Yu-Kwong K., Ishfaq A. Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors // ACM Computing Surveys. 1999. 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 Bag-of-Tasks Applications // IEEE Transactions on Parallel and Distributed Systems. 2008. Vol. 19, no. 5. P. 698–709. DOI: 10.1109/TPDS.2007.70747.

5. Xia Y., Prasanna V. Hierarchical Scheduling of DAG Structured Computations on Manycore Processors with Dynamic Thread Grouping // Job Scheduling Strategies for Parallel Processing. Springer, 2010. Lecture Notes in Computer Science. Vol. 6253, 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 21st Annual Symposium on Principles of Distributed Computing, PODC '02, Monterey, California, July 21–24, 2002. New York, ACM, 2002. P. 280–289. DOI: 10.1145/571825.571876.

7. Hendler D., Shavit N. Work Dealing // Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '02, Winnipeg, Canada, August 10–13, 2002. New York, ACM, 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 // ACM SIGPLAN Notices. 2013. 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 10th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '98, Puerto Vallarta, Mexico, June 28 – July 2, 1998. New York, ACM, 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. Vol. 46, no. 2. P. 173–197. DOI: 10.1007/s10766-016-0484-8.

11. Blumof R.D., Joerg C.F., Kuszmaul B.C., *et al.* Cilk: An Efficient Multithreaded Runtime System // ACM SIGPLAN Notices. 1995. Vol. 30, no. 8. P. 207–216. DOI: 10.1145/209937.209958.

12. Leijen D., Schulte W., Burckhardt S. The Design of a Task Parallel Library // ACM SIGPLAN Notices. 2009. Vol. 44, no. 10. P. 227–242. DOI: 10.1145/1639949.1640106.

13. Tardieu O., Wang H., Lin H. A Work-Stealing Scheduler for X10's Task Parallelism with Suspension // Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '12, New Orleans, Louisiana, USA, February 25–29, 2012. New York, ACM, 2012. P. 267–276. DOI: 10.1145/2145816.2145850.

14. Knuth D. The Art of Computer Programming. Volume 1. Addison-Wesley Professional, 1997. 672 p.

15. Blumofe R.D., Leiserson C.E. Scheduling Multithreaded Computations by Work Stealing // Journal of the ACM. 1999. Vol. 46, no. 5. P. 720–748. DOI: 10.1145/324133.324234.

16. 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.

17. 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.

18. Li J., Agrawal K., Elnikety S., *et al.* Work Stealing for Interactive Services to Meet Target Latency // ACM SIGPLAN Notices. 2016. Vol. 51, no. 8. P. 1–13. DOI: 10.1145/3016078.2851151.

19. Robison A., Voss M., Kukanov A. Optimization via Reflection on Work Stealing in TBB // Proceedings of the IEEE International Parallel & Distributed Processing Symposium, IPDPS '08, Miami, FL, USA, April 14–18, 2008. IEEE, 2008. P. 1–8. DOI: 10.1109/IPDPS.2008.4536188.

20. Dijk T., Pol J.C. Lace: Non-blocking Split Deque for Work-Stealing // Parallel Processing Workshops. Springer, 2014. Lecture Notes in Computer Science. Vol. 8806, P. 206–217. DOI: 10.1007/978-3-319-14313-2_18.

21. Faxén K.-F. Wool-A work stealing library // ACM SIGARCH Computer Architecture News. 2008. Vol. 36, no. 5. P. 93–100. DOI: 10.1145/1556444.1556457.

22. 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: 10.1002/cpe.3771.

23. Wimmer M., Versaci F., Traff J.L., *et al.* Data Structures for Task-based Priority Scheduling // ACM SIGPLAN Notices. 2014. Vol. 49, no. 8. P. 379–380. DOI: 10.1145/2692916.2555278.

24. Chen Q., Guo M., Guan H. LAWS: Locality-Aware Work-Stealing for Multi-socket Multi-core Architectures // Proceedings of the 28th ACM International Conference on

Supercomputing, ICS'14, Munich, Germany, June 10–13, 2014. New York, ACM, 2014. P. 3–12. DOI: 10.1145/2597652.2597665.

25. Guo H., Chen Q., Guo M., Xu L. SAWS: Selective Asymmetry-Aware Work-Stealing for Asymmetric Multi-core Architectures // Proceedings of the IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, Sydney, Australia, December 12–14, 2016. IEEE, 2016. P. 116–123. DOI: 10.1109/HPCC-SmartCity-DSS.2016.0027.

26. Guo Y., Zhao J., Cave V., Sarkar V. SLAW: A Scalable Locality-aware Adaptive Work-stealing Scheduler for Multi-core Systems // ACM SIGPLAN Notices. 2010. Vol. 45, no. 5. P. 341–342. DOI: 10.1145/1837853.1693504.

27. Mitzenmacher M. Analyses of Load Stealing Models Based on Differential Equations // Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA/PODC98, Puerto Vallarta, Mexico, June 28 – July 2, 1998. New York, ACM, 1998. P. 212–221. DOI: 10.1145/277651.277687.

28. Wangkai J., Xiangjun P. SLITS: Sparsity-Lightened Intelligent Thread Scheduling // Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '23, Orlando, Florida, United States, June 19–23, 2023. New York, ACM, 2023 P. 21–22. DOI: 10.1145/3578338.3593568.

29. Кучумов Р.И. Реализация и анализ Work-Stealing планировщика задач // Стохастическая оптимизация в информатике. 2016. Т. 12, № 1. С. 20–39.

30. Sokolov A.V., Drac A.V. The Linked List Representation of n LIFO-Stacks and/or FIFO-Queues in the Single-Level Memory // Information Processing Letters. 2013. Vol. 113, no. 19–21. P. 832–835. DOI: 10.1016/j.ipl.2013.07.021.

31. 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.

32. Hendler D., Lev Y., Moir M., Shavit N. A Dynamic-Sized Nonblocking Work Stealing Deque // Distributed Computing. 2006. Vol. 18. P. 189–207. DOI: 10.1007/s00446-005-0144-5.

33. Соколов А.В. Математические модели и алгоритмы оптимального управления динамическими структурами данных. Петрозаводск: Изд-во ПетрГУ, 2002. 215 с.

34. Драц А.В., Соколов А.В. Управление двумя FIFO-очередями в случае их движения друг за другом по кругу // Математические методы распознавания образов. 2011. Т. 15, № 1. С. 315–317.

35. Drac A.V., Sokolov A.V. The Circular Representation of 2 FIFO-Queues in Single Level Memory // Proceedings of the International Conference on Numerical Analysis and Applied Mathematics, ICNAAM-2014, Rhodes, Greece, September 22–28, 2014. AIP Publishing LLC, 2015. Vol. 1648. P. 520002. DOI: 10.1063/1.4912732.

36. Барковский Е.А., Соколов А.В. Способ управления памятью компьютерной системы (RU 2647627 C1). URL: `http://www1.fips.ru/registers-doc-view/fips_servlet?DB=RUPAT&DocNumber=2647627&TypeFile=html` (дата обращения: 18.07.2023).

37. Sokolov A., Barkovsky E. The Mathematical Model and the Problem of Optimal Partitioning of Shared Memory for Work-Stealing Deques // Parallel Computing Technologies (PaCT 2015): Proceedings of the 13th International Conference, Petrozavodsk, Russia, August 31 – September 4, 2015. Springer, 2015. Lecture Notes in Computer Science. Vol. 9251, P. 102–106. DOI: 10.1007/978-3-319-21909-7_11.

38. 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.

39. Aksenova E.A., Sokolov A.V. Modeling of the Memory Management Process for Dynamic Work-Stealing Schedulers // Proceedings of Ivannikov ISPRAS Open Conference, ISPRAS 2017, Moscow, Russia, November 30 – December 1, 2017. IEEE, 2017. P. 12–15. DOI: 10.1109/ISPRAS.2017.00009.

40. 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.

41. 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. Vol. 40. P. 1763–1770. DOI: 10.1134/S1995080219110052.

42. 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.

43. Work-Stealing Task Scheduler. URL: `https://github.com/rkuchumov/staccato`.

44. CPP Distributed Scheduler. URL: `https://gitlab.com/mildlyparallel/cpp-distributed-scheduler`.

45. Хайдарова Р.Р., Муромцев Д.И., Лапаев М.В., Фищенко В.Д. Модель распределенной сверточной нейронной сети на кластере компьютеров с ограниченными вычислительными ресурсами // Научно-технический вестник информационных технологий, механики и оптики. 2020. Т. 20. № 5. С. 739–746. DOI: 10.17586/2226-1494-2020-20-5-739-746.

46. Saddik A., Latif R., Ouardi A.E., *et al.* Computer development based embedded systems in precision agriculture: tools and application // Acta Agriculturae Scandinavica, Section B – Soil & Plant Science. 2022. Vol. 72, no. 1. P. 589–611. DOI: 10.1080/09064710.2021.2024874.

47. 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.

48. Аксёнова 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.

49. 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. Vol. 42. P. 1475–1482. DOI: 10.1134/S1995080221070027.

50. Aksenova E., Sokolov A. Minimizing the Average Cost of Redistribution when Working with Two Work-Stealing Deques in Two-Level Memory. Russian Supercomputing Days: Proceedings of Russian Supercomputing Days, Moscow, Russia, September 26–27, 2022. Moscow, MAKS Press, 2022. P. 4–12. URL: https://russianscdays.org/files/2022/RuSCDays22_Proceedings.pdf

Аксёнова Елена Алексеевна, к.ф.-м.н., н.с., Институт прикладных математических исследований КарНЦ РАН (Петрозаводск, Российская Федерация)

Соколов Андрей Владимирович, д.ф.-м.н., в.н.с., Институт прикладных математических исследований КарНЦ РАН (Петрозаводск, Российская Федерация)

DOI: 10.14529/cmse230403

CONTROL METHODS OF WORK-STEALING DEQUES IN DYNAMIC SCHEDULERS OF MULTIPROCESSOR PARALLEL COMPUTATIONS

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

Received: 21.07.2023

In parallel task schedulers, which are using the work-stealing strategy, each processor has own task deque. One end of the deque is used for insertion and deletion of tasks only by the owner, and the other is used for stealing of tasks by other processors. The article offers an overview of work-stealing deque's description of the deque's optimal management problems, which our team had solved for the work-stealing strategy. The idea of the algorithm for deque's managing in two-level memory is that if the memory allocated to the deques becomes overflow, elements are redistributed between memory levels. Elements from the deque's ends are stored in fast memory, since they will be worked with in the near time, and elements from the deque's middle part are stored in slow memory. In this case, it is necessary to determine the required number of elements that need to be left in fast memory, depending on the optimal criteria and system parameters.

Keywords: controlled random walks, optimal control of work-stealing deques, optimal deque caching, optimization of work-stealing load schedulers, simulation and Markov models of optimal control of data structures.

FOR CITATION

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

References

1. Herlihy M., Shavit N. The Art of Multiprocessor Programming. Elsevier, 2008. 508 p.
2. Yu-Kwong K., Ishfaq A. Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors. ACM Computing Surveys. 1999. 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 Bag-of-Tasks Applications. IEEE Transactions on Parallel and Distributed Systems. 2008. Vol. 19, no. 5. P. 698–709. DOI: 10.1109/TPDS.2007.70747.
5. Xia Y., Prasanna V. Hierarchical Scheduling of DAG Structured Computations on Manycore Processors with Dynamic Thread Grouping. Job Scheduling Strategies for Parallel Processing. Springer, 2010. Lecture Notes in Computer Science. Vol. 6253, 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 21st annual symposium on Principles of distributed computing, PODC '02, Monterey, California, July 21–24, 2002. New York, ACM, 2002. P. 280–289. DOI: 10.1145/571825.571876.
7. Hendler D., Shavit N. Work Dealing. Proceedings of the 14th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '02, Winnipeg, Canada, August 10–13, 2002. New York, ACM, 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. ACM SIGPLAN Notices. 2013. 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 10th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '98, Puerto Vallarta, Mexico, June 28 – July 2, 1998. New York, ACM, 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. Vol. 46, no. 2. P. 173–197. DOI: 10.1007/s10766-016-0484-8.
11. Blumof R.D., Joerg C.F., Kuszmaul B.C., *et al.* Cilk: An Efficient Multithreaded Runtime System. ACM SIGPLAN Notices. 1995. Vol. 30, no. 8. P. 207–216. DOI: 10.1145/209937.209958.
12. Leijen D., Schulte W., Burckhardt S. The Design of a Task Parallel Library. ACM SIGPLAN Notices. 2009. Vol. 44, no. 10. P. 227–242. DOI: 10.1145/1639949.1640106.
13. Tardieu O., Wang H., Lin H. A Work-Stealing Scheduler for X10's Task Parallelism with Suspension. Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '12, New Orleans, Louisiana, USA, February 25–29, 2012. New York, ACM, 2012. P. 267–276. DOI: 10.1145/2145816.2145850.
14. Knuth D. The Art of Computer Programming. Volume 1. Addison-Wesley Professional, 1997. 672 p.

15. Blumofe R.D., Leiserson C.E. Scheduling Multithreaded Computations by Work Stealing. Journal of the ACM. 1999. Vol. 46, no. 5. P. 720–748. DOI: 10.1145/324133.324234.

16. 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.

17. 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.

18. Li J., Agrawal K., Elnikety S., *et al.* Work Stealing for Interactive Services to Meet Target Latency. ACM SIGPLAN Notices. 2016. Vol. 51, no. 8. P. 1–13. DOI: 10.1145/3016078.2851151.

19. Robison A., Voss M., Kukanov A. Optimization via Reflection on Work Stealing in TBB. Proceedings of the IEEE International Parallel & Distributed Processing Symposium, IPDPS '08, Miami, FL, USA, April 14–18, 2008. IEEE, 2008. P. 1–8. DOI: 10.1109/IPDPS.2008.4536188.

20. Dijk T., Pol J.C. Lace: Non-blocking Split Deque for Work-Stealing. Parallel Processing Workshops. Springer, 2014. Lecture Notes in Computer Science. Vol. 8806, P. 206–217. DOI: 10.1007/978-3-319-14313-2_18.

21. Faxén K.-F. Wool-A work stealing library. ACM SIGARCH Computer Architecture News. 2008. Vol. 36, no. 5. P. 93–100. DOI: 10.1145/1556444.1556457.

22. 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: 10.1002/cpe.3771.

23. Wimmer M., Versaci F., Traff J.L., *et al.* Data Structures for Task-based Priority Scheduling. ACM SIGPLAN Notices. 2014. Vol. 49, no. 8. P. 379–380. DOI: 10.1145/2692916.2555278.

24. Chen Q., Guo M., Guan H. LAWS: Locality-Aware Work-Stealing for Multi-socket Multi-core Architectures. Proceedings of the 28th ACM International Conference on Supercomputing, ICS'14, Munich, Germany, June 10–13, 2014. New York, ACM, 2014. P. 3–12. DOI: 10.1145/2597652.2597665.

25. Guo H., Chen Q., Guo M., Xu L. SAWS: Selective Asymmetry-Aware Work-Stealing for Asymmetric Multi-core Architectures. Proceedings of the IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, Sydney, Australia, December 12–14, 2016. IEEE, 2016. P. 116–123. DOI: 10.1109/HPCC-SmartCity-DSS.2016.0027.

26. Guo Y., Zhao J., Cave V., Sarkar V. SLAW: A Scalable Locality-aware Adaptive Work-stealing Scheduler for Multi-core Systems. ACM SIGPLAN Notices. 2010. Vol. 45, no. 5. P. 341–342. DOI: 10.1145/1837853.1693504.

27. Mitzenmacher M. Analyses of Load Stealing Models Based on Differential Equations. Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA/PODC98, Puerto Vallarta, Mexico, June 28 – July 2, 1998. New York, ACM, 1998. P. 212–221. DOI: 10.1145/277651.277687.

28. Wangkai J., Xiangjun P. SLITS: Sparsity-Lightened Intelligent Thread Scheduling. Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '23, Orlando, Florida, United States, June 19–23, 2023. New York, ACM, 2023 P. 21–22. DOI: 10.1145/3578338.3593568.

29. Kuchumov R.I. Implementation and Analysis of the Work-Stealing Task Scheduler. Stochastic Optimization in Computer Science. 2016. Vol. 12, no. 1. P. 20–39. (in Russian).

30. Sokolov A.V., Drac A.V. The Linked List Representation of n LIFO-Stacks and/or FIFO-Queues in the Single-Level Memory. Information Processing Letters. 2013. Vol. 113, no. 19–21. P. 832–835. DOI: 10.1016/j.ipl.2013.07.021.

31. Aksenova E.A., Lazutina A.A., Sokolov A.V. On Optimal Methods of Representing Dynamic Data Structures. Review of Applied and Industrial Mathematics. 2003. Vol. 10, no. 2. P. 375–376. (in Russian).

32. Hendler D., Lev Y., Moir M., Shavit N. A Dynamic-Sized Nonblocking Work Stealing Deque. Distributed Computing. 2006. Vol. 18. P. 189–207. DOI: 10.1007/s00446-005-0144-5.

33. Sokolov A.V. Mathematical Models and Algorithms of Optimal Control of Dynamic Data Structures. PetrSU Publishing, 2002. 215 p. (in Russian).

34. Drac A.V., Sokolov A.V. Control of Two FIFO-Queues in the Case of Their Movement One After Another in a Circle. Mathematical Methods of Pattern Recognition. 2011. Vol. 15, no. 1. P. 315–317. (in Russian).

35. Drac A.V., Sokolov A.V. The Circular Representation of 2 FIFO-Queues in Single Level Memory. Proceedings of the International Conference on Numerical Analysis and Applied Mathematics, ICNAAM-2014, Rhodes, Greece, September 22–28, 2014. AIP Publishing LLC, 2015. Vol. 1648. P. 520002. DOI: 10.1063/1.4912732.

36. Barkovsky E.A., Sokolov A.V. A Method of Memory Control of a Computer System (RU 2647627 C1). URL: `http://www1.fips.ru/registers-doc-view/fips_servlet?DB=RUPAT&DocNumber=2647627&TypeFile=html` (in Russian).

37. Sokolov A., Barkovsky E. The Mathematical Model and the Problem of Optimal Partitioning of Shared Memory for Work-Stealing Deques. Parallel Computing Technologies (PaCT 2015): Proceedings of the 13th International Conference, Petrozavodsk, Russia, August 31 – September 4, 2015. Springer, 2015. Lecture Notes in Computer Science. Vol. 9251, P. 102–106. DOI: 10.1007/978-3-319-21909-7_11.

38. 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. DOI: 10.25209/2079-3316-2017-8-1-83-103. (in Russian).

39. Aksenova E.A., Sokolov A.V. Modeling of the Memory Management Process for Dynamic Work-Stealing Schedulers. Proceedings of Ivannikov ISPRAS Open Conference, ISPRAS 2017, Moscow, Russian Federation, November 30 – December 1, 2017. IEEE, 2017. P. 12–15. DOI: 10.1109/ISPRAS.2017.00009.

40. 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: 10.25209/2079-3316-2019-10-1-3-17.

41. 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. Vol. 40. P. 1763–1770. DOI: 10.1134/S1995080219110052.

42. 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.

43. Work-Stealing Task Scheduler. URL: `https://github.com/rkuchumov/staccato`.

44. CPP Distributed Scheduler. URL: `https://gitlab.com/mildlyparallel/cpp-distributed-scheduler`.

45. Khaydarova R.R., Mouromtsev D.I., Lapaev M.V., Fishchenko V.D. Distributed convolutional neural network model on resource-constrained cluster. Scientific and Technical Journal of Information Technologies, Mechanics and Optics, 2020, Vol. 20, no. 5, P. 739–746. DOI: 10.17586/2226-1494-2020-20-5-739-746.

46. Saddik A., Latif R., Ouardi A.E., *et al.* Computer development based embedded systems in precision agriculture: tools and application. Acta Agriculturae Scandinavica, Section B – Soil & Plant Science. 2022. Vol. 72, no. 1. P. 589–611. DOI: 10.1080/09064710.2021.2024874.

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

48. Aksenova E.A, Lazutina A.A., Sokolov A.V. Minimizing the average cost of redistribution when working with work-stealing deques in two-level memory. Program Systems: Theory and Applications. 2021. Vol. 12, no. 2. P. 53–71. DOI: 10.25209/2079-3316-2021-12-2-53-71. (in Russian).

49. 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. Vol. 42. P. 1475–1482. DOI: 10.1134/S1995080221070027.

50. Aksenova E., Sokolov A. Minimizing the Average Cost of Redistribution when Working with Two Work-Stealing Deques in Two-Level Memory. Russian Supercomputing Days: Proceedings of Russian Supercomputing Days, Moscow, Russia, September 26–27, 2022. Moscow, MAKS Press, 2022. P. 4–12. URL: `https://russianscdays.org/files/2022/RuSCDays22_Proceedings.pdf`