УДК 004.272.44

КООПЕРАТИВНАЯ ПОТОКОВАЯ МОДЕЛЬ РЕШЕНИЯ ЗАДАЧ БОЛЬШОЙ РАЗМЕРНОСТИ НА МНОГОЯДЕРНЫХ СИСТЕМАХ

А.А. Прихожий
Профессор кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» Белорусского национального технического университета профессор, доктор технических наук

О.Н. Карасик
Аспирант Белорусского национального технического университета

Белорусский национальный технический университет, Республика Беларусь
E-mail: prihozhy@yahoo.com

Аннотация. Рассмотрена проблема решения задач большой размерности на многоядерных системах. Предложена кооперативная модель разработки блочно-параллельных алгоритмов и управления выполнением многопоточных приложений. Она оптимизирует порядок выполнения операций и обмен данными, уменьшает общее время выполнения многопоточных приложений посредством сокращения критических путей на графе параллельного алгоритма, увеличивает пропускную способность приложения при росте числа потоков и исключает конкуренцию потоков, назначенных на один процессор.

Ключевые слова: задача большого размера, кооперативная модель, блочно-параллельный алгоритм, многопоточное приложение, многоядерная система.

Введение. Повсеместное использование многоядерных систем делает возможным решение разнообразных прикладных задач большой размерности посредством создания эффективных многопоточных параллельных приложений. Разработка многопоточного приложения, способного эффективно задействовать весь потенциал многоядерной системы, является трудоемкой задачей, Широкое распространение получают целевые библиотеки, платформы и компиляторы, которые ускоряют процесс разработки многопоточных приложений, сокращают объем работ, выполняемых программистом, предоставляют более простые и удобные средства управления потоками. Нашей целью явилась разработка модели и средств улучшения параметров многопоточного приложения, решающего конкретную прикладную задачу.

В области параллельных систем предложен ряд моделей вычислительных процессов: сетевые алгоритмы [1,2], конвейерные вычислительные планы [3-4], преобразование последовательного алгоритма к одноблочной потоковой модели [5] и др. Важнейшим инструментом анализа и экстракции параллелизма являются средства оценки критического пути в программном коде алгоритма [6]. Модели многопоточных приложений исследовались в работах [7-12].

Кооперативная потоковая модель. Понятие кооперативной модели потоковых блочно-параллельных алгоритмов, которое исследуется в данной статье, отличается от понятия кооперативной многозадачности в операционной системе. Потоковый блочно-параллельный алгоритм разрабатывается для каждой конкретной решаемой задачи, учитывает ее специфику и является средством реализации кооперативной многопоточности на многоядерных процессорах. Он обеспечивает параллельное высокопроизводительное решение вычислительной

задачи, декомпозируемой на части. В основе построения модели лежат следующие принципы.

Принцип 1. Один поток размещает один блок данных алгоритма. Блок данных может представлять сегмент вектора, блок матрицы, подграф графа и т.д. Он может быть структурированным и составным.

Принцип 2. Потоки группируются на ядрах многоядерной системы. Разбиение потоков на группы и назначение групп на ядра должно балансировать вычислительную нагрузку на ядрах с учетом их производительности и должно сокращать обмен данными между ядрами.

Принцип 3. Загрузка ядер полезными вычислениями, описанными в блочно-параллельном алгоритме и решающими поставленную задачу, должна приближаться к 100%, а затраты времени на управление потоками, а также слоты времени ожидания и простаивания должны быть минимальными.

Принцип 4. Потоки, назначенные на одно ядро, выполняются последовательно, а вычисления, описанные в них, сериализуются не смотря на то, что потенциально они могли быть выполнены, пусть даже частично, параллельно.

Принцип 5. Потоки, назначенные на разные ядра, выполняются параллельно-последовательно. Порядок их выполнения и синхронизации с другими потоками определяется зависимостями по данным между блоками и между потоками. При назначении блоков и потоков на разные ядра необходимо стремиться к тому, чтобы они могли выполняться минимально последовательно и максимально параллельно.

Принцип 6. Исходный порядок выполнения потоков, назначенных на одно ядро и определяемый зависимостями по данным является частичным порядком. Частичный порядок доопределяется до полного порядка в процессе построения и оптимизации блочно-параллельного алгоритма так, чтобы слоты времени простаивания в ожидании готовности необходимых данных, вычисляемых на других ядрах, были минимальными.

Принцип 7. Уменьшение размера блока приводит к более плотной упаковке блоков по оси времени на каждом из ядер и приводит к уменьшению слотов времени простаивания ядер в ожидании готовности данных, необходимых для вычисления очередного блока.

Принцип 8. Существует предельное уменьшение размера блока, за которым общее время работы блочно-параллельного алгоритма начинает возрастать из-за накладных расходов, связанных с частыми передачами управления и частыми посылками данных от одного потока другому потоку.

Принцип 9. Нахождение оптимального размера блока является важнейшим источником оптимизации потокового блочно-параллельного алгоритма.

Кооперативный планировщик потоков. Предлагаются средства планирования, повышающие загрузку ядер системы и сокращающие обмен данными между ядрами. Их разработка основана на механизме User-Mode Scheduling (UMS), который доступен, в частности, в ОС семейства Windows. На базе API UMS строится и реализуется кооперативный планировщик, позволяющий осуществлять передачу управления от одного потока другому, определять порядок выполнения пользовательских потоков без участия системного планировщика. Разработаны три версии кооперативного планировщика: A0, A1 и A2. Базовая архитектура A0 планировщика построена непосредственно средствами API UMS и состоит из менеджера памяти, пользовательских потоков и потоков планировщика.

Модифицированная архитектура A1 расширяет архитектуру A0 посредством разработки нового примитива синхронизации с целью исключения взаимодействия потоков пользователя с ОС в процессах блокировки-разблокировки, а также посредством добавления своей очереди заблокированных потоков в каждый поток планировщика. Усовершенствованная архитектура A2 расширяет архитектуру A1 посредством изменения механизма работы с очередью потоков, а также посредством модификации и ускорения процессов блокировки и разблокировки пользовательских потоков за счет переноса соответствующей логики из потока

планировщика в примитив синхронизации. Эффективность планировщика возрастает от A0 к A1 и от A1 к A2. На рис.1 показан выигрыш усовершенствованного планировщика по отношению к базовому на двух потоковых кооперативных алгоритмах μ1κ и μ2κ.

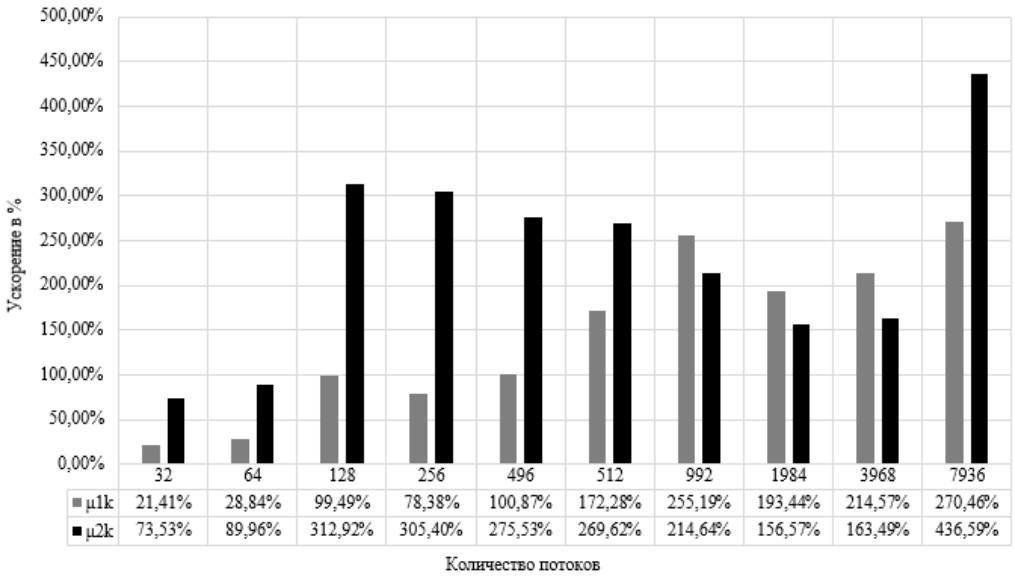


Рисунок 1. Сокращение в процентах общего времени выполнения обратного хода алгоритмами μ1κ и μ2κ для усовершенствованного кооперативного планировщика по сравнению с базовым планировщиком в зависимости от количества потоков

Потоковые кооперативные блочно-параллельные алгоритмы решения СЛАУ. Кооперативная потоковая модель позволила разработать на основе двух блочных методов Гаусса два потоковых блочно-параллельных алгоритма μ1κ и μ2κ. В условиях балансирования нагрузки на логические процессоры, основным принципом работы алгоритмов является минимизация числа передач управления между потоками на одном процессоре и сокращение критического пути в потоковом алгоритме. Алгоритмы представляется высокоуровневыми конечными автоматами, работающими в нескольких режимах. Показано, что алгоритмы сокращают критический путь, а также повышает коэффициент распараллеленности и загрузку процессоров по сравнению с методом Гаусса. Сравнение алгоритмов μ1κ и μ2κ между собой показывает, что μ1κ порождает меньше передач управления между потоками, однако μ2κ дает более короткий критический путь и большую загрузку процессоров.

На рис.2 показаны графики ускорения многопоточных приложений, реализующих блочно-параллельные алгоритмы μ1k и μ2k и их блочно-параллельные прототипы μ1 и μ2, над однопоточным приложением, реализующим последовательный метод Гаусса, в зависимости от числа потоков. Анализ графиков выполнен в контексте используемой экспериментальной среды, включающей 16 логических процессоров и 8 физических ядер с использованием технологии Hyper-Threading. Наихудшие результаты показал алгоритм μ2 с ускорением 5.12x в среднем по всем конфигурациям потоков.. Причинами этого являются использование планировщика с вытесняющей многозадачностью, а также увеличние обмена данными между ядрами и между уровнями кэш памяти. Алгоритм μ1, показавший заметное увеличение ускорения (с 4.76x до 10.41x) с ростом числа потоков, превосходит алгоритм μ2. Ускорение μ1 над однопоточным методом Гаусса составило 6.36x в среднем по всем конфигурациям потоков. Максимальное ускорение 10.41x получено для 1984 потоков. Ускорение μ1k и μ2k над однопоточным методом Гаусса составило 7.38x и 7.50x в среднем

соответственно. Разработанные потоковые алгоритмы μ1k и μ2k превосходят известные алгоритмы μ1 и μ2 в среднем на 29.62 %. На 3968 потоках алгоритмы μ1k и μ2k показали близкие максимальные ускорения 12.67x и 12.60x соответственно.

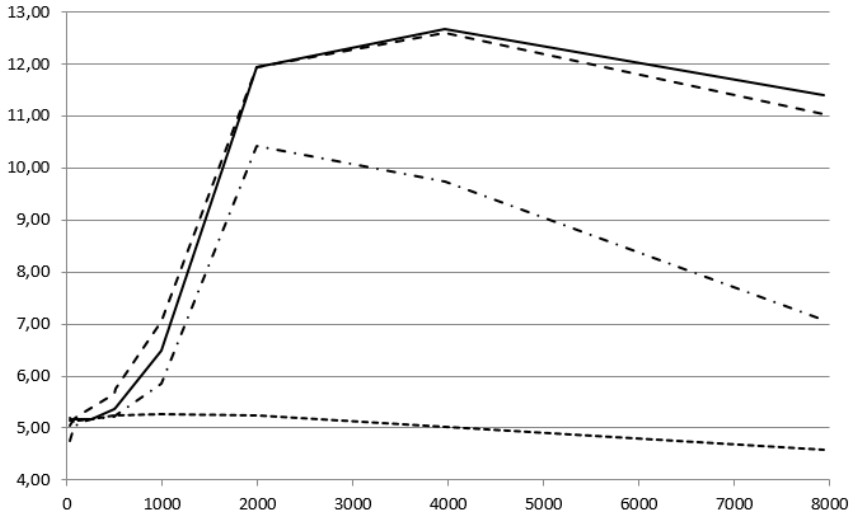
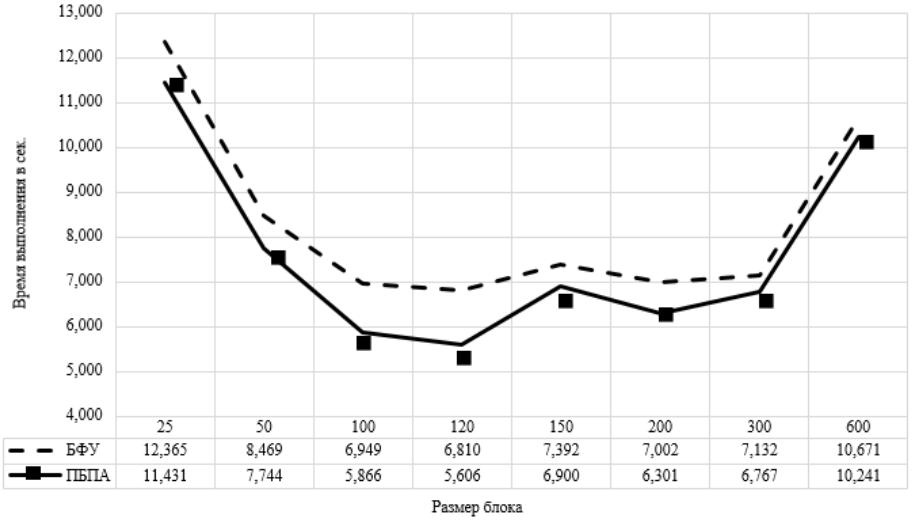


Рисунок 2. Ускорение в разах параллельных алгоритмов μ1k (сплошная) и μ2k (пунктирная), реализованных на усовершенствованном кооперативном планировщике, и параллельных алгоритмов μ1 (штрихпунктирная) и μ2 (точечная), реализованных планировщиком вытесняющей многозадачности, по сравнению с лучшим однопоточным приложением в зависимости от числа потоков для СЛАУ размером 15872 переменных

Потоковый кооперативный блочно-параллельный алгоритм поиска кратчайших путей на графе. Предлагается новый потоковый блочно-параллельный алгоритм (ПБПА), который разбивает граф на подграфы, распределяет множество подграфов (блоков) по потокам и разбивает множество потоков по процессорам, локализуя данные и вычисления внутри потоков и процессоров. Он изменяет порядок вычисления блоков по сравнению с блочным алгоритмом Флойда-Уоршелла (БФУ) таким образом, что увеличивает загрузку ядер и сокращает обмен данными между кэш памятью отдельных ядер.

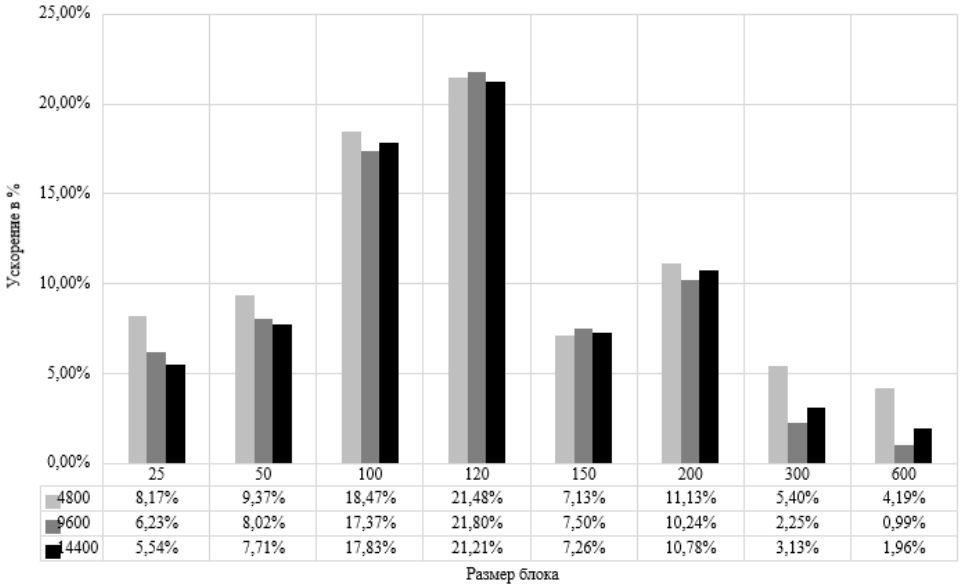
Зависимость времени выполнения алгоритмов на многоядерной системе от размера блока показана на рис.3.



	25	50	100	120	150	200	300	600
БФУ	12,365	8,469	6,949	6,810	7,392	7,002	7,132	10,671
ПБПА	11,431	7,744	5,866	5,606	6,900	6,301	6,767	10,241

Рисунок 3. Зависимость CPU-времени поиска кратчайших путей алгоритмами БФУ (пунктирная линия) и ПБПА (сплошная линия) от размера блока для графа из 4800 вершим

Сопоставление алгоритмов по производительности дает рис.4.



	25	50	100	120	150	200	300	600
4800	8,17%	9,37%	18,47%	21,48%	7,13%	11,13%	5,40%	4,19%
9600	6,23%	8,02%	17,37%	21,80%	7,50%	10,24%	2,25%	0,99%
14400	5,54%	7,71%	17,83%	21,21%	7,26%	10,78%	3,13%	1,96%

Рисунок 4. Сокращение времени работы алгоритма ПБПА по сравнению с алгоритмом БФУ в зависимости от размера блока для трех размеров графа: 4800, 9600 и 14400 вершин

Предлагаемый алгоритм ПБПА выигрывает у известного алгоритма БФУ более 21% при оптимальном размере блока. Увеличение производительности обусловлено сокращением критического пути в параллельном алгоритме ПБПА, которое достигнуто изменением порядка вычисления блоков. ПБПА обладает свойством масштабируемости при настройке на различные многоядерные архитектуры а также с случае увеличения числа ядер.

Список литературы

[1]. Prihozhy A. Asynchronous Scheduling and Allocation // DATE 98 – Design, Automation and Test in Europe: Proc. European Conf., Paris, France. – 1998. – P. 934-935.

[2]. Prihozhy A., Mlynek D., Solomennik M., Mattavelli M. Techniques for Optimization of Net Algorithms // Proc. Int. Conf. “PARELEC’2002 – Parallel Computing in Electrical Engineering”, IEEE CS, Los Alamitos, California. – 2002. – P. 211-216.

[3]. Prihozhy A., Bezati E., Rahman H., Mattavelli M. Synthesis and Optimization of Pipelines for HW Implementations of Dataflow Programs // IEEE Trans. on CAD of Integrated Circuits and Systems. – 2015. – Vol. 34, No. 10. – P. 1613-1626.

[4]. Прихожий А.А., Ждановский А.М., Карасик О.Н., Маттавелли М. Эвристический генетический алгоритм оптимизации вычислительных конвейеров // Доклады БГУИР. – 2017. – № 1. – С. 34-41.

[5]. Прихожий А.А., Соломенник М. Распараллеливание и планирование вычислитель-ных и информационных процессов // Доклады БГУИР. – 2003. – № 4. – С. 104-114.

[6]. Prihozhy A., Mattavelli M., Mlynek D. Data Dependences Critical Path Evaluation at C/C++ System Level Description // Part of book “Integrated Circuit and System Design”, LNCS 2799, Springer. – 2003. – P. 569-579.

[7]. Прихожий А.А., Карасик О.Н. Исследование методов реализации многопоточных приложений на многоядерных системах // Информатизация образования. – 2014. – № 1. – С. 43-62.

[8]. Прихожий А.А., Карасик О.Н. Кооперативная модель оптимизации выполнения потоков на многоядерной системе // Системный анализ и прикладная информатика. – 2014. – №4. – С. 13-20.

[9]. Прыхожы А.А., Карасік А.М. Кааператыўныя блочна-паралельныя алгарытмы рашэння задач на шмат'ядравых сістэмах // Сістэмны аналіз і прыкладная інфарматыка. – 2015. – №2. – С. 10-18.

[10]. Карасик О.Н., Прихожий А.А. Усовершенствованный планировщик кооперативного выполнения потоков на многоядерной системе // Системный анализ и прикладная математика. – 2017. – № 1. – С. 4-11.

[11]. Прихожий А.А., Карасик О.Н. Разнородный блочный алгоритм поиска кратчайших путей между всеми парами вершин графа // Системный анализ и прикладная информатика. – №3. – 2017. – С. 68-75.

[12]. Карасик О.Н., Прихожий А.А. Потоковый блочно-параллельный алгоритм поиска кратчайших путей на графе // Доклады БГУИР. – 2018. – № 2. – С. 77-84.

COOPERATIVE THREADED MODEL OF SOLVING LARGE-SIZE PROBLEMS ON MULTI-CORE SYSTEMS

A.A. PRIHOZHY,
Doctor of Engeneering Sciences
Professor professor at the Computer and System Software Department of the Belarusian National Technical University

A.N. KARASIK
Postgraduate student Belarusian National Technical University

Belarusian National Technical University, Republic of Belarus
E-mail: prihozhy@yahoo.com

Abstract. The problem of solving large-size tasks on multi-core systems is considered. The cooperative model for the development of threaded block-parallel algorithms and for the management of multi-threaded applications execution has been proposed. It optimizes the order of operations execution and data exchanges, decreases the overall time of the multi-threaded applications execution by means of the reduction of critical paths in the concurrent algorithm graph, increases the application throughput at the growth of the number of threads, and excludes the competition among threads.

Key words: Large-size task, cooperative model, block-parallel algorithm, multi-threaded application, multi-core system.