Russian Technological Journal.
2022;10(6):7–19

ISSN 2782-3210 (Print)
ISSN 2500-316X (Online)

УДК 004.4'2 + 004.51
https://doi.org/10.32362/2500-316X-2022-10-6-7-19

НАУЧНАЯ СТАТЬЯ

Вычислительная сложность построения рациональных планов выполнения программ на заданном поле параллельных вычислителей

В.М. Баканов @

МИРЭА – Российский технологический университет, Москва, 119454 Россия
@ *Автор для переписки, e-mail: e881e@mail.ru*

Резюме

Цели. Построение рациональных планов (расписаний) выполнения параллельных программ (ВПП), вследствие неоднозначности, является сложной задачей. Цель работы – создание методик разработки таких планов и специализированного программного обеспечения для реализации этих методик, полагающихся на внутренние свойства алгоритмов, в первую очередь на свойство внутреннего (скрытого) параллелизма.

Методы. Основными методами при разработке планов ВПП являются построение, анализ и целенаправленное преобразование ярусно-параллельной формы (ЯПФ) информационных графов алгоритмов (ИГА). Преобразование ЯПФ осуществляется путем переноса операторов с яруса на ярус ЯПФ (именно это событие и принято за элементарный шаг при определении вычислительной сложности выполнения сценария). В качестве инструмента преобразования применен метод разработки сценариев преобразования на скриптовом языке программирования Lua. Сценарии создаются на основе эвристического подхода и используют набор API-функций (API – Application Programming Interface) разработанной программной системы, позволяющих всесторонне изучить параметры ИГА и его ЯПФ-представления для последующего построения плана ВПП на заданном поле параллельных вычислителей.

Результаты. Результаты вычислительных экспериментов выявили особенности внутренних свойств алгоритмов, влияющих на эффективность преобразований ЯПФ. Получены сравнительные показатели вычислительной сложности получения планов ВПП и иных параметров (включая плотность кода и др.) при применении различных сценариев преобразования ЯПФ. Итерационный подход к улучшению эвристических методов позволит приблизиться к оптимальным схемам решения целевой задачи.

Выводы. В целом разработанный программный комплекс подтвердил эффективность в исследовании параметров скрытого параллелизма в произвольных алгоритмах и рационального его использования при обработке данных. Подход применения скриптового языка для разработки эвристических методов (сценариев) целенаправленного преобразования форм ИГА показал большую гибкость и прозрачность для исследователя. Целевыми потребителями разработанных методов генерации расписаний параллельного выполнения программ в первую очередь являются разработчики трансляторов и виртуальных машин, исследователи свойств алгоритмов (в направлении нахождения и использования потенциала скрытого их параллелизма). Разработанное программное обеспечение и методики несколько лет применяются при обучении студентов в университетах России, что позволило повысить компетенции учащихся в области параллелизации обработки данных.

Ключевые слова: граф алгоритма, тонкая информационная структура программы, ярусно-параллельная форма графа, рациональные параметры выполнения параллельной программы, план выполнения параллельной программы

• **Поступила:** 30.01.2022 • **Доработана:** 29.03.2022 • **Принята к опубликованию:** 05.09.2022

Для цитирования: Баканов В.М. Вычислительная сложность построения рациональных планов выполнения программ на заданном поле параллельных вычислителей. *Russ. Technol. J.* 2022;10(6):7–19. https://doi.org/10.32362/2500-316X-2022-10-6-7-19

Прозрачность финансовой деятельности: Автор не имеет финансовой заинтересованности в представленных материалах или методах.

Автор заявляет об отсутствии конфликта интересов.

RESEARCH ARTICLE

Computational complexity when constructing rational plans for program execution in a given field of parallel computers

Valery M. Bakanov @

MIREA – Russian Technological University, Moscow, 119454 Russia
@ *Corresponding author, e-mail: e881e@mail.ru*

Abstract

Objectives. The construction of rational plans (schedules) for parallel program execution (PPE) represents a challenging problem due to its ambiguity. The aim of this work is to create methods for developing such plans and specialized software for implementing these methods, which are based on the internal properties of algorithms, primarily on the property of internal (hidden) parallelism.

Methods. The main method for developing PPE plans was the construction, analysis, and purposeful transformation of the stacked-parallel form (SPF) of information graphs of algorithms (IGA). The SPF was transformed by transferring operators from tier to tier of the SPF (this event was taken as an elementary step in determining the computational complexity of scenario execution). As a transformation tool, a method for developing transformation scenarios in the scripting programming language Lua was used. Scenarios were created by a heuristic approach using a set of Application Programming Interface (API) functions of the developed software system. These functions formed the basis for a comprehensive study of the parameters of the IGA and its SPF representation for the subsequent construction of a PPE plan applying to a given field of parallel computers.

Results. Features of the internal properties of the algorithms that affect the efficiency of SPF transformations were identified during the course of computational experiments. Comparative indices of the computational complexity of obtaining PPE plans and other parameters (including code density, etc.) were obtained for various SPF transformation scenarios. An iterative approach to improving heuristic methods favors developing optimal schemes for solving the objective problem.

Conclusions. The developed software system confirmed its efficiency for studying the parameters of hidden parallelism in arbitrary algorithms and rational use in data processing. The approach of using a scripting language to develop heuristic methods (scenarios) for the purposeful transformation of IGA forms showed great flexibility and transparency for the researcher. The target consumers of the developed methods for generating schedules for parallel execution of programs are, first of all, developers of translators and virtual machines, and researchers of the properties of algorithms (for identifying and exploiting the potential of their hidden parallelism). The developed software and methods have been successfully used for a number of years for increasing student competence in data processing parallelization at Russian universities.

Keywords: algorithm graph, fine information structure of program, stacked-parallel form of graph, rational execution parameters of parallel program, execution plan of parallel program

• **Submitted:** 30.01.2022 • **Revised:** 29.03.2022 • **Accepted:** 05.09.2022

For citation: Bakanov V.M. Computational complexity when constructing rational plans for program execution in a given field of parallel computers. *Russ. Technol. J*. 2022;10(6):7–19. https://doi.org/10.32362/2500-316X-2022-10-6-7-19

Financial disclosure: The author has no a financial or property interest in any material or method mentioned.

The author declares no conflicts of interest.

ВВЕДЕНИЕ

В настоящее время для сокращения времени вычислений широко применяется параллелизация, реализуемая путем одновременной обработки данных по частям на множестве различных вычислительных устройств с последующим объединением полученных результатов. Параллельное выполнение позволяет «обойти» существующие фундаментальные и чисто технологические ограничения на рост производительности путем повышения тактовой частоты процессоров. Однако задача выявления (обычно скрытого для непосвященного наблюдателя) потенциала параллелизма в алгоритмах не является «лежащей на поверхности», а уж эффективность использования параллелизма – тем более [1].

При организации параллельных вычислений отдельной и совсем непростой проблемой становится получение плана (расписания) выполнения параллельной программы (ПП). Дело в том, что практически каждую последовательную программу (алгоритмы, являющиеся базовыми компонентами любой программы, традиционно принято представлять в последовательном виде) можно представить в параллельной форме множеством способов при сохранении инварианта в виде набора операций и причинно-следственной связи между ними, причем каждый способ будет обладать различной эффективностью выполнения на параллельном вычислителе заданной архитектуры. При такой постановке задачи возрастает ценность изучения внутренних свойств именно алгоритмов[1] как реальных блоков построения программ.

Каждый из возможных планов выполнения ПП ассоциирован с определенными параметрами качества выполнения программы (время, требуемые вычислительные ресурсы, загрузка памяти и др.). При этом, в зависимости от поставленной задачи, возможны формулировка целевой функции и решение оптимизационной многопараметрической задачи.

[1] Воеводин В., Донгарра Дж. (ред.). *AlgoWiki. Открытая энциклопедия свойств алгоритмов*. http://algowiki-project.org. Дата обращения 15.01.2022. [Voevodin V., Dongarra J. (Eds.). *AlgoWiki: Open Encyclopedia of Parallel Algorithmic Features*. http://algowiki-project.org. Accessed October 20, 2022 (in Russ.).]

Несмотря на различия в архитектуре и системах машинных команд различных параллельных вычислительных систем и технологий параллельного программирования, целесообразно иметь научно обоснованные общие подходы к построению укрупненных планов (расписаний) выполнения программ. При абстрагировании от конкретики технологий параллельного программирования такие планы логично называть *каркасом* выполнения параллельной программы.

В работе предложены методы и их реализация (в форме программных сценариев решения поставленной задачи) для разработки рациональных планов выполнения ПП на заданном поле параллельных вычислителей, включая гетерогенное. Разработанные сценарии предназначены для встраивания в качестве распараллеливающих блоков во вновь разрабатываемые системы создания исполняемого кода программ. При этом упор делается именно на достижение максимального быстродействия, т.к. процессы отладки программ требуют многократной трансляции со сложной оптимизацией при условии достижения необходимого качества.

МЕТОДЫ

Для решения задачи нахождения методов построения рациональных планов выполнения ПП создана специализированная программная система инструментального уровня, обобщенная схема и схема информационных потоков которой приведены на рис. 1.

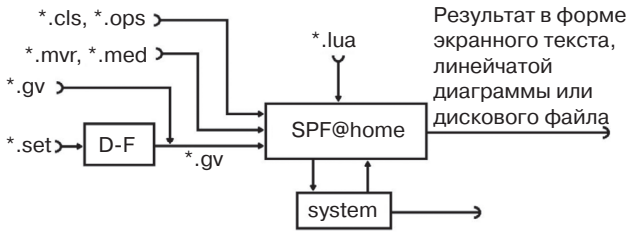


Рис. 1. Схема инструментального программного комплекса для построения планов выполнения параллельных программ [2]

На вход программной системы (рис. 1) поступает описание анализируемого алгоритма в форме традиционной последовательной программы в императивном стиле или формального ее описания в виде

ориентированного ациклического информационного графа алгоритма (ИГА) – зависимость вида «операторы → операнды». При этом вершины графа ассоциируются с операторами (группами операторов) программы, а дуги – с линиями передачи данных. На этом рисунке типы файлов *.set и *.gv – соответственно программный файл и файл информационного графа анализируемой программы; *.mvr, *.med – соответственно файлы метрик вершин и дуг графа алгоритма; *.cls, *.ops – соответственно файлы параметров вычислителей и операторов программы; *.lua – текстовый файл на языке Lua, содержащий сценарии получения планов выполнения параллельных программ. При использовании результатов разработки в качестве алгоритмической основы функционирования компонентов распараллеливающих трансляторов ИГА строится на основе результатов парсинга исходного кода транслятором (при этом библиотечные функции обрабатываются отдельно). При реализации приложения в виде отдельного компонента *Code Morphing Software* ИГА строится на основе сгенерированного последовательного исполняемого кода. Полезным свойством использования представления программ в виде ИГА является возможность обработки графа отдельными блоками (уровня сабрутин, файлов) с последующей сборкой графа полного приложения.

Выявление и анализ внутреннего логического параллелизма в алгоритмах реализовано с помощью имитации модели а́кторов (модуль D-F) и построения специальных сечений ИГА в виде его ярусно-параллельной формы (ЯПФ) [3] (модуль SPF@home) [4, 5]. Оба упомянутых модуля разработаны с использованием языка C/C++ в стиле GUI для модели Win’32 (для проведения массовых расчетов дополнительно реализован режим работы с командной строкой), являются полностью open source и могут быть выгружены для свободного использования (формат инсталляционных файлов)[2]. Модуль D-F строит план выполнения ПП для модели асинхронизма выполнения операторов, расписания в стиле модуля SPF@home рассчитаны на синхронный вызов групп операторов.

Модуль D-F (Data-Flow) является фактически универсальным вычислителем архитектуры SMP (*Symmetric MultiProcessing*, системы с общей памятью, [6]), на вход которого подается последовательная программа на императивном ассемблероподобном языке (используются 3-символьные мнемоники команд и 3-адресная система с порядком следования операндов согласно соглашениям AT & T). Программа выполняется в симуляторе вычислителя статической архитектуры Data-Flow. При этом порядок выполнения инструкций процессора определяется не порядком их следования в машинном коде, а готовностью операндов [7, 8].

Возможность варьирования числа параллельных вычислителей и правил выборки команд из буфера [4] позволяет решать обратную задачу оптимизации параметров вычислителя по характеристикам процесса вычислений. Предусмотрен экспорт ИГА в сторонние программы в DOT-формате, ведется подробный протокол симуляции.

Условное выполнение реализуется предикатным методом [9, 10], для реализации циклов используется система макросов, «разворачивающих» циклические структуры. Удобству визуализации решения способствует выдача данных о процессе выполнения программ в виде функции интенсивности вычислений (ИВ, число одновременно выполнившихся операторов в функции времени) и диаграмм Ганта.

Модуль SPF@home [5] предназначен для моделирования и выбора наилучших (в заданном смысле) сценариев преобразования ЯПФ как плана параллельного выполнения операторов на вычислительной системе заданной архитектуры. Существенным преимуществом использования ЯПФ является удовлетворительная временна́я вычислительная сложность ее получения (квадратично-полиномиальная относительно количества вершин графа). ЯПФ изначально может быть построена в «верхнем» или «нижнем» вариантах (все операторы располагаются на максимально приближенных к началу или концу выполнения программы ярусах ЯПФ соответственно). Важным пользовательским преимуществом ЯПФ является большая наглядность представления выявленного параллелизма в алгоритме – даже первоначально полученная ЯПФ алгоритма уже является некоторым начальным (обычно далеким от оптимального) планом выполнения ПП.

При этом расположенные на ярусах ЯПФ группы (связки, *bundles*) операторов выполняются параллельно в пределах каждого яруса, а выполнение групп операторов на каждом ярусе происходит последовательно, начиная с начального яруса. Минимально возможная высота ЯПФ (число ярусов) определяется длиной критического пути в ИГА [11, 12] и обуславливает кратчайшее время выполнения алгоритма. В модуле SPF@home визуализация полученной ЯПФ осуществляется в текстовом и графическом виде (линейчатая диаграмма ширин ЯПФ позволяет без напряжения представить вид функции распределения ширин ЯПФ по ее высоте). Механизм задания метрик выполнения операторов позволяет задавать параметры для операторов

[2] http://vbakanov.ru/dataflow/content/install_df.exe. Дата обращения 24.10.2022. / Accessed October 24, 2022 (in Russ.); http://vbakanov.ru/spf@home/content/install_spf.exe. Дата обращения 24.10.2022. / Accessed October 24, 2022 (in Russ.).

(например, время их выполнения или требуемые ресурсы при гетерогенном поле вычислителей), вычислителей (типы операторов, которые могут быть выполнены на данном вычислителе) и линий передачи данных (время передачи, размеры данных). Практика использования механизма метрик подробно описана в подразделе, посвященном построению расписания выполнения ПП программ на заданном поле гетерогенных вычислителей.

Дополнительно модуль SPF@home предоставляет возможность получить информацию о требуемом для выполнения заданного алгоритма *времени жизни данных*. Существование этих данных является следствием выполнения отдельных операторов и, в свою очередь, они служат входными операндами для иных операторов алгоритма. Эта информация (фактически оценка локальной *емкостной сложности* выполнения программы) важна для определения требуемых параметров внутренних регистров процессора и/или решения вопросов об оптимальности размещения данных между регистрами процессора и оперативной памятью.

Основным методом преобразования ЯПФ служит целенаправленное перемещение операторов между ярусами ЯПФ при сохранении информационных связей в ИГА. В целом данная задача относится к классу задач построения расписаний с ограничениями и является *NP*-полной [13]. Вследствие *NP*-полноты задачи можно говорить о построении именно *рациональных* (итеративно стремящихся к оптимальным) планов выполнения параллельных программ. В данной работе используется эвристический подход к решению, причем сценарии решения (в данном случае – преобразования ЯПФ в нужном направлении) реализуются с использованием скриптового языка Lua [14]. Lua выбран по причине полного open source, близости синтаксиса к распространенным языкам программирования (стиль C/C++), компактности при встраивании в родительское приложение.

Каждый вызов Lua-функции фактически является оберткой над соответствующим API[3]-вызовом родительской программы. Набор API системы SPF@home охватывает практически все обозримые действия над ИГА и позволяет реализовать анализ графа любой (ограниченной ресурсами вычислителя) сложности. В этом смысле использование ЯПФ графа является лишь одним из возможных приемов решения. Всего можно различить три типа вызовов:

- информационные (служат для получения информации об ИГА и его ЯПФ; на основании этих данных в дальнейшем выбирается конкретный метод обработки ИГА для решения поставленной задачи). Примеры – получить общее число ярусов ЯПФ, число операторов на заданном ярусе, диапазон возможного расположения данного оператора по ярусам ЯПФ и др.;
- акционные (служат для реализации конкретных методов решения задачи построения расписания выполнения ПП). Примеры – построить «верхнюю» или «нижнюю» форму ЯПФ, добавить пустой ярус под данным, перенести оператор с яруса на ярус и др.;
- вспомогательные (вывод рассчитанных данных в текстовом и графическом виде для обмена данными с иными приложениями, работа с файловой системой и т.п.).

Информационный граф алгоритма в виде ЯПФ формально может быть представлен 2D-списком из элементов-идентификаторов (например, уникальных номеров) операторов $[a_{ij}]$, где $i = 1 \ldots W$ – номер строки; величина W называется высотой ЯПФ и определяется величиной критического пути в ИГА; $j = 1 \ldots j_i^{\max}$ – число операторов на строке i. Величина $H = \max_i(1 \ldots j_i^{\max})$ называется шириной ЯПФ. В реальном ИГА положение каждого оператора на ярусах ограничено наличием информационных связей в алгоритме и ограничено диапазоном $i^{\min} \le i \le i^{\max}$ (где $i^{\min}$ и $i^{\max}$ – допустимые номера ярусов размещения данного оператора в ЯПФ; диапазон $i^{\min} – i^{\max}$ логично назвать *вариативностью* положений по ярусам i-го оператора). ЯПФ фактически является (исходным, наивным) планом (расписанием) выполнения ПП. При заданном описании направление орта времени совпадает с увеличением номера яруса ЯПФ.

В целом предлагаемый подход полностью соответствует стилю EPIC (Explicitly Parallel Instruction Computing, набор инструкций с явным параллелизмом [15]) и предназначен для программной реализации распараллеливающих блоков трансляторов. При этом для вычислительной архитектуры VLIW (Very Long Instruction Word, сверхдлинная машинная команда [10]) под термином «оператор» следует понимать именно машинную команду (здесь имеем полное следование концепции ILP – Instruction-Level Parallelism, параллелизм на уровне команд [4]). Для многопоточных систем на многоядерных процессорах «оператор» логично соотнести с *гранулой параллелизма* значительно большего размера, например, уровня оператора/операторов или процедур языка программирования высокого уровня [16]. Последний вариант хорошо укладывается в концепцию интерпретаторов. В обоих случаях представленная общая методика построения рационального плана выполнения ПП остается неизменной.

Внутренняя реализация данных, конечно, совсем не обязана предусматривать явного построения ЯПФ в виде 2D-массива. Она может быть любой удобной для компьютерной реализации, например,

[3] API – Application Programming Interface.

в наивном случае – устанавливающей однозначное соответствие между ИГА в виде множества направленных дуг $\{k, l\}$ (матрица смежности), идентифицированных парами номеров вершин i_k, j_k и i_l, j_l, где i, j – номера строк и столбцов в ЯПФ.

В качестве образцов для исследования использовались широко распространенные алгоритмы обработки данных (линейная алгебра, статистика, операции над массивами и др.). Дополнительно были подготовлены искусственные (не соответствующие ни одному из применяемых алгоритмов, но сгенерированные в согласии с заданными параметрами) ИГА. Вполне определенным недостатком экспериментального материала являются относительно небольшие размерности обрабатываемых данных. Это связано со значительной трудностью ручного составления программ. Однако проведенные эксперименты показывают усиление выявленных тенденций при повышении размерности обрабатываемых данных во всем исследованном диапазоне их изменения.

Для определения вычислительной сложности выполнения сценариев преобразования ЯПФ использован аналог классического метода оценки целевого параметра, используемый при операциях сортировки массивов – определение количества *элементарных шагов* (перестановки двух элементов сортируемого массива), необходимых для завершения операции. В нашем случае за элементарный шаг логично принять перестановку оператора с яруса на ярус ЯПФ. Такой подход обладает всеми достоинствами и недостатками классического метода, включая неучет сложности анализа ситуации и принятия решений о совершении конкретного элементарного шага.

В данном исследовании также оценивалась величина *плотности кода*, характеризующая степень использования ресурсов параллельной вычислительной системы (числа вычислителей) при выполнении данного алгоритма (формально, отклонение ширин ярусов преобразованной ЯПФ от заданного значения). При неполном использовании вычислительных ресурсов транслятор вынужденно вставляет инструкции NOP на «пустые» места в связках параллельно выполняемых команд, что приводит к понижению эффективности кода.

Нижеприведенная серия экспериментов проводилась с использованием модуля SPF@home, как обладающего наибольшей гибкостью в преобразовании ЯПФ графов алгоритмов; собственно ИГА генерировались модулем D-F на основе программного кода. В ходе вычислительных экспериментов модуль SPF@home сохраняет подробнейший протокол моделирования для последующего анализа. Для данной работы представляют особый интерес следующие параметры, полученные в ходе целевого преобразования ЯПФ:

- высота H и ширина W полученной ЯПФ (ограничения на ширину задавались в качестве параметра, обеспечивающего выполнение алгоритма на заданном числе параллельных вычислителей);
- равномерность распределения ширин ярусов (*плотность кода*) в данной ЯПФ оценивалась величиной коэффициента вариации $CV = \frac{1}{\bar{W}}\sqrt{\frac{\sum(W-\bar{W})^2}{H-1}}$, где $\bar{W}$ – среднеарифметическое ширин ярусов по ЯПФ;
- вычислительная сложность выполненного преобразования ЯПФ (в единицах числа перестановок операторов с яруса на ярус ЯПФ).

Для сравнения эффективности преобразования ЯПФ использовали два (представленные в виде Lua-скриптов) эвристических метода (сценария), основанных на различных общих подходах к преобразованию ЯПФ. Первый (условное название *01_Strategy*) использует основанный на способе дихотомии подход массового переноса операторов с яруса на ярус ЯПФ, второй (*02_Strategy*) – постепенный перенос операторов с «более нагруженных» на «менее нагруженные» ярусы. В обоих случаях при необходимости в нужных местах создаются дополнительные (изначально пустые) ярусы для обеспечения выполнения алгоритма на заданном числе параллельных вычислителей.

РЕЗУЛЬТАТЫ

Построение расписания выполнения программ на фиксированном числе параллельных вычислителей при возможности увеличения времени выполнения программы

В этом подразделе рассматривается наиболее общий случай, соответствующий условию $\bar{W} >> P$, где P – число параллельных вычислителей. Именно при этом приходится увеличивать высоту ЯПФ (время выполнения программы).

Эффективность вышеописанных эвристических методов проверялась путем последовательного их применения к ЯПФ исследуемых алгоритмов в диапазоне числа параллельных вычислителей P от W_0 (ширина исходной ЯПФ) до 1 (полностью последовательное выполнение алгоритма).

На рис. 2–5 показано изменение целевых величин (оси ординат): (а) – вычислительной сложности, (б) – высоты ЯПФ, (в) – коэффициента вариации ширин ярусов ЯПФ, как функций заданного количества параллельных вычислителей P (оси абсцисс). Преобразования ЯПФ соответствующих алгоритмов проводились согласно сценариям *01_Strategy* и *02_Strategy* (кривые 1 и 2 соответственно).

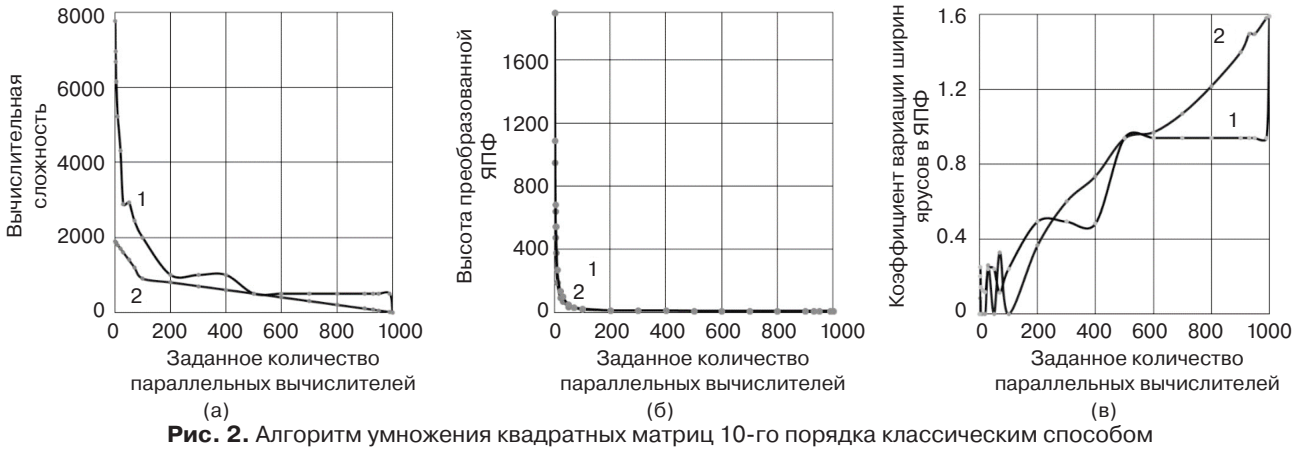


Рис. 2. Алгоритм умножения квадратных матриц 10-го порядка классическим способом

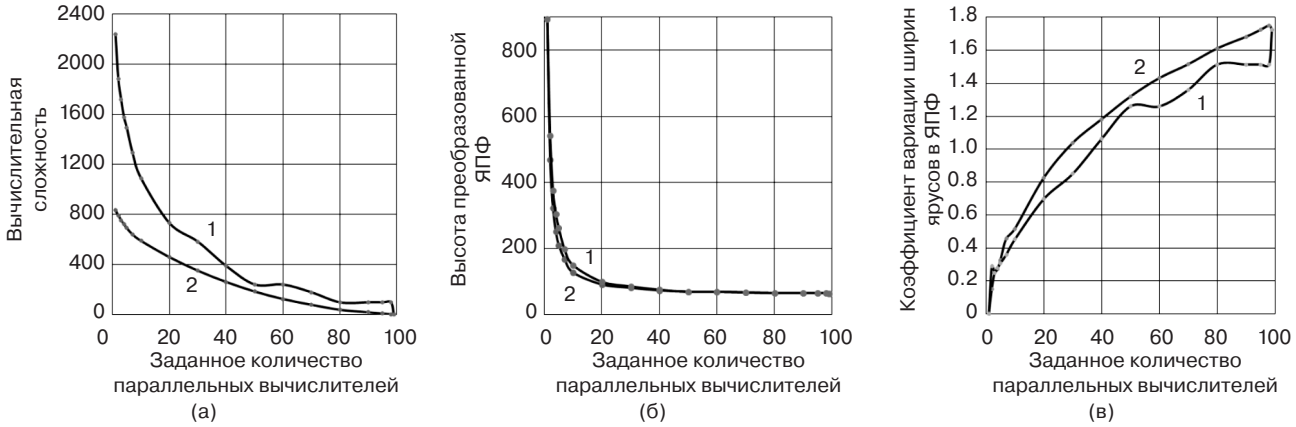


Рис. 3. Алгоритм решения систем линейных алгебраических уравнений (СЛАУ) 10-го порядка прямым (безитерационным) методом Гаусса

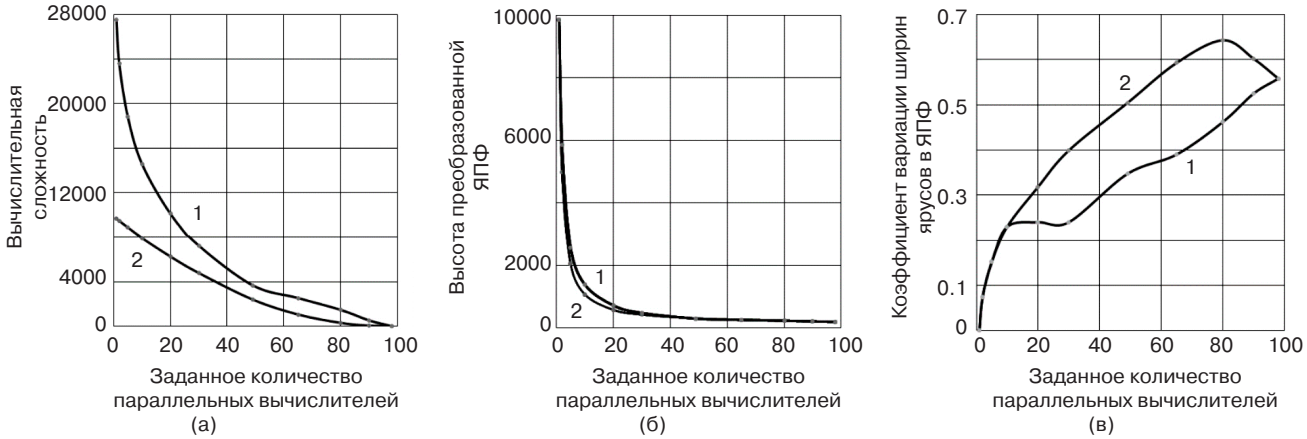


Рис. 4. Искусственно сгенерированный алгоритм e19039_o9853_t199

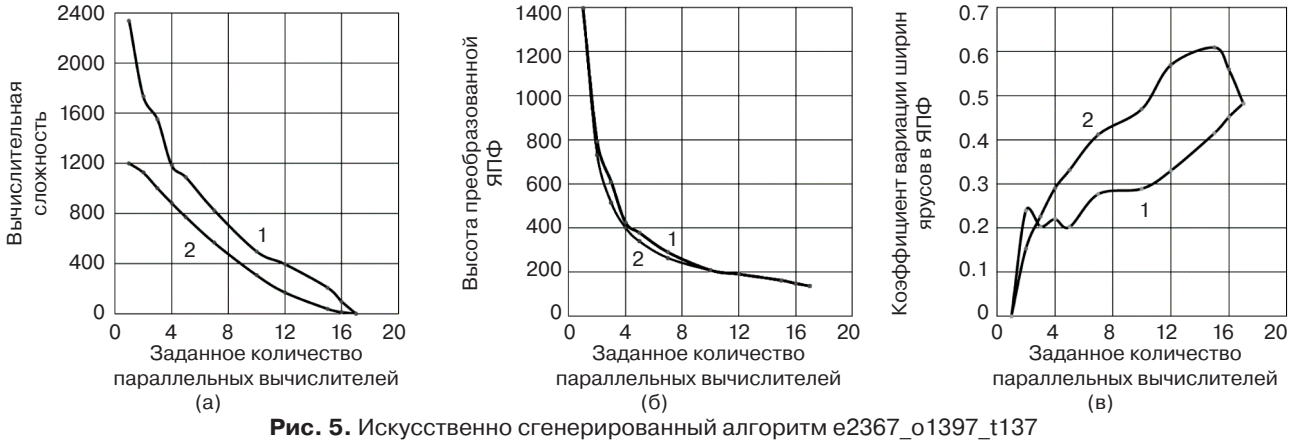


Рис. 5. Искусственно сгенерированный алгоритм e2367_o1397_t137

Как видно из рис. 2–5, оба метода на исследованных алгоритмах приводят к близким результатам – сценарий *02_Strategy* более «быстрый» относительно антагониста (графики (а)) при малоотличимом времени выполнения алгоритма (графики (б)). По плотности кода (графики (в)) оба сценария показывают сходные тенденции стремления к минимуму целевой функции при небольшом (существенно меньшем ширины ЯПФ) числе параллельных вычислителей. Причудливые формы кривых являются следствием сложности используемых сценариев и обработкой целочисленных величин, причем при использовании сценария *02_Strategy* графики визуально обладают более монотонной формой.

Количественно сценарий *02_Strategy* имеет меньшую (приблизительно в 2–4 раза в исследуемом диапазоне размеров обрабатываемых данных) вычислительную сложность относительно *01_Strategy*, хотя на первый взгляд ожидалось противоположное. Однако при этом сценарий *02_Strategy* обладает более сложной внутренней логикой по сравнению с *01_Strategy* (в последнем случае она примитивна), что не может быть учтено принятой системой оценки вычислительной сложности. С учетом сказанного может быть логичнее использовать в компонентах распараллеливающих систем сценарий *01_Strategy* для быстрого, но достаточно «грубого» построения планов выполнения параллельных программ, а метод *02_Strategy* – для построения этих планов в режиме оптимизации.

Построение расписания выполнения программ на минимальном числе параллельных вычислителей при условии невозрастания времени выполнения программы

В случае $\bar{W} \approx P$ (среднеарифметическое значений ширин исходной ЯПФ сравнимо с числом параллельных вычислителей) возникает задача получения расписания выполнения ПП с максимальной плотностью кода без возрастания высоты ЯПФ («балансировка» операторов по ярусам ЯПФ). Разработанные эмпирические методы по «балансировке» ЯПФ показали противоречивые результаты – в некоторых случаях удавалось достичь почти 100% плотности кода, а некоторые алгоритмы практически не поддавались указанной модификации (вследствие ограничений на перемещения операторов между ярусами из-за необходимости сохранения информационных зависимостей в алгоритме).

Применение программного модуля SPF@home возможно и для решения обратной задачи, например, определения параметров параллельной вычислительной системы исходя из баланса производительности и стоимости собственно системы.

Перспективные методы расчета расписания выполнения параллельных программ

Все перечисленные эксперименты характерны тем, что ЯПФ первоначально вычислялась в «верхней» форме (все операторы находились как можно «выше» по ярусам ЯПФ). В этом случае преимущественными перемещениями операторов по ярусам были направления «сверху вниз» – от начальных ярусов к конечным (это не запрещает, конечно, использование «многоходовок» с неоднократным изменением направления перемещения операторов между ярусами).

Описанная последовательность действий логически обоснована характерной формой кривых ИВ в условиях неограниченного параллелизма – резкий подъем в начальной части сменяется пиком и плавным снижением к концу выполнения программы. Такая форма кривой возникает при значительных объемах входных данных и лишний раз подтверждает принадлежность процесса выполнения операторов на поле параллельных вычислителей к одной из разновидностей многоканальных систем массового обслуживания. Так как при этом вариативность находящихся в области пика функции ИВ операторов велика, эффективность получения удовлетворительного плана выполнения ПП путем перемещения операторов ЯПФ «вниз» также значительна. Будем называть алгоритмы, начальная (нереформированная) форма распределения ширин которых в «верхней» ЯПФ соответствует вышеописанной, принадлежащими к П-классу[4].

В порядке обсуждения небезынтересно будет рассмотреть вариант начальной ЯПФ в «нижней» форме (при этом все операторы перемещены максимально в сторону окончания выполнения программы). Такая ЯПФ может быть получена из «верхней» перемещениями операторов по ярусам «как можно ниже» или много проще – построением ЯПФ в направлении от конца программы к ее началу (в последнем случае вычислительная сложность получения ЯПФ остается такой же, как и при получении «верхней» ЯПФ). Ниже проиллюстрировано сравнение распределения ширин ЯПФ в «верхней» и «нижней» формах.

[4] К П-классу будем относить алгоритмы, которые характеризуются распределением ширин ярусов в ЯПФ информационного графа в «верхней» форме (когда все операторы находятся как можно ближе к началу выполнения) с наличием ярко выраженного максимума в начале и пологого снижения в конце. [We will refer to the П class such algorithms that are characterized by the distribution of the tier widths in the SPF of the information graph in the upper form (when all operators are as close as possible to the beginning of execution) with the presence of a pronounced maximum at the beginning and of a gentle decrease at the end.]

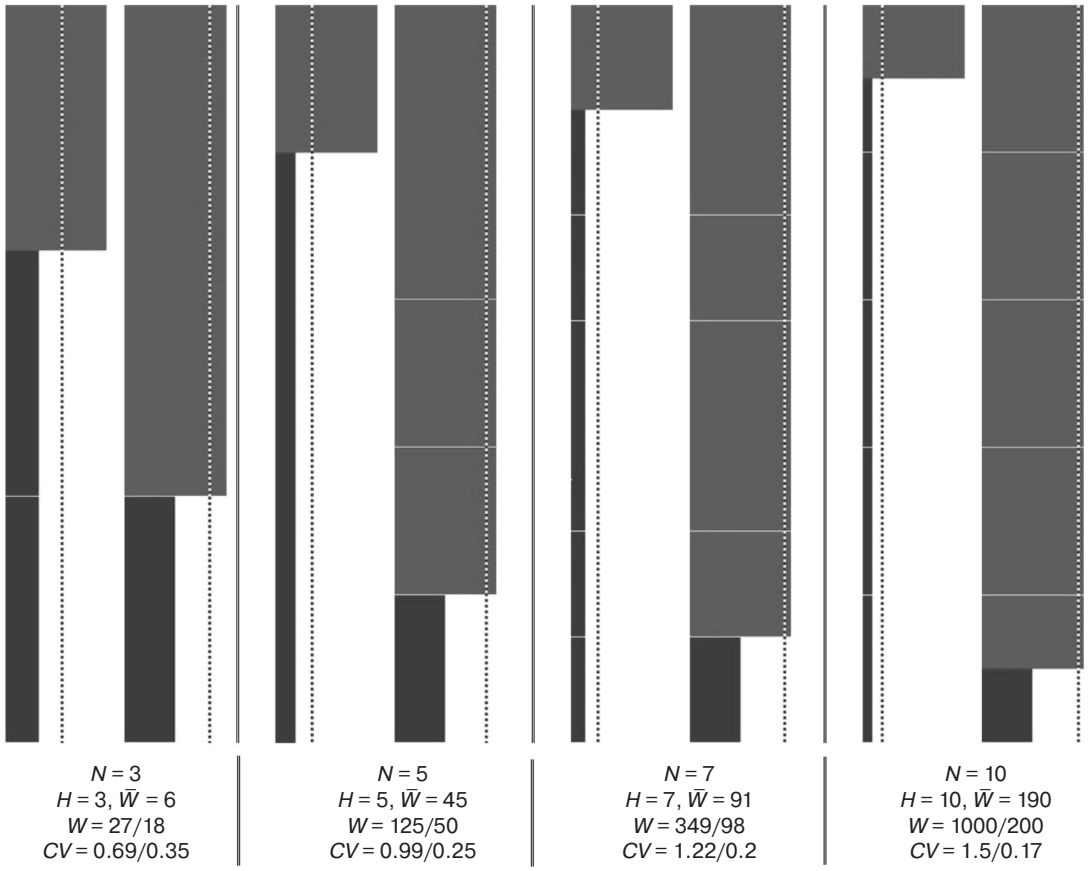


Рис. 6. Алгоритм умножения квадратных матриц классическим способом; горизонтальная ось – ширина ярусов ЯПФ для порядка матриц $N = 3, 5, 7, 10$. Светло-серый цвет – ярусы максимальной ширины, темно-серый – минимальной ширины

На рис. 6 приведена линейчатая диаграмма распределения ширин ЯПФ для заданных размеров обрабатываемых данных (указаны в подрисуночных надписях), полученная модулем SPF@home, причем в каждом из 4 столбцов строки приведены две диаграммы – слева для «верхней» и справа для «нижней» ЯПФ данного алгоритма. Здесь H, W и $\overline{W}$ – высота, ширина и среднеарифметическая ширина ЯПФ (последняя показана на рисунке пунктиром; символ прямого слеша разделяет параметры для «верхней» и «нижней» ЯПФ).

Последние представленные иллюстрации интересны возможностью существенной «балансировки» ЯПФ без применения сложных эвристических алгоритмов ее реорганизации. Фактически все возможные решения при реорганизации ЯПФ находятся в диапазоне между «верхней» и «нижней» формами, однако при выборе в качестве исходной «нижней» формы приоритетным перемещением операторов является движение «вверх».

Последнее высказывание требует ответа на вопрос – не имеются ли среди двух пограничных («верхней» и «нижней») форм состояний с еще лучшим показателем «балансировки»? Для ответа на этот вопрос был проведен эксперимент по пошаговому преобразованию ЯПФ из «верхней» формы в «нижнюю» (на рис. 7 оцифровка оси абсцисс соответствует числу перемещений операторов с яруса на ярус), в дополнение к показателю CV (сплошные линии, левая шкала ординат) неравномерность ширин ярусов ЯПФ оценивалась отношением ширины самого широкого яруса к самому узкому (пунктир, правая шкала ординат).

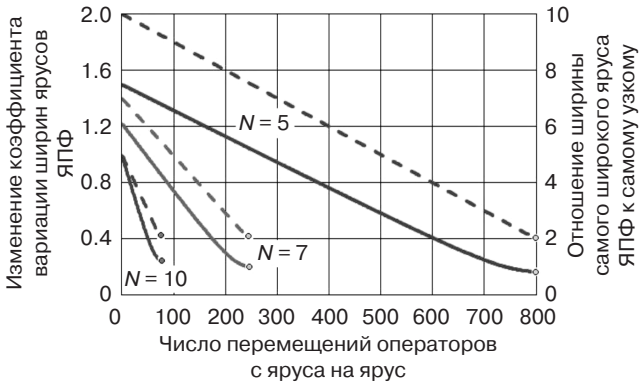


Рис. 7. Неравномерность распределения ширин ярусов ЯПФ для алгоритма умножения матриц классическим способом (N – порядок перемножаемых матриц)

Несмотря на то, что в данных экспериментах не имело места полное покрытие всего множества состояний ЯПФ (возможные перемещения операторов по ярусам ЯПФ совершались «вниз» на максимальную вариативность), можно с высокой степенью вероятности утверждать, что целевые величины обладают свойством невозрастания в диапазоне существования и минимальны именно в области «нижней» ЯПФ.

В случае нахождения легко определяемых условий принадлежности алгоритма к П-классу возможна существенная экономия вычислений при определении рациональных планов выполнения ПП – вместо реализации вышеописанных достаточно сложных сценариев достаточно построить «нижнюю» ЯПФ. Использованные ранее количественные характеристики неравномерности ширин ярусов не дают информации о форме кривой, обладающей этой неравномерностью. В качестве дополнительной оценки неравномерности распределения операторов по ярусам ЯПФ может быть использован известный графоаналитический метод определения дифференциации доходов населения[5], заключающийся в расчете численных параметров расслоения (кривая Лоренца и коэффициент Джини), несмотря на зеркально-противоположную форму анализируемых кривых.

Пример показывает важность изучения свойств (включая классификацию) алгоритмов со стороны их сущностной грани, представляющей собой внутренний параллелизм, с целью наилучшего практического использования этих свойств.

Построение расписания выполнения параллельных программ на заданном числе гетерогенных вычислителей

Современные многоядерные процессоры все чаще разрабатываются с вычислительными ядрами различных возможностей. Поэтому практически полезно уметь строить расписание выполнения параллельных программ для подобных систем (с гетерогенным полем параллельных вычислителей).

Модуль SPF@home поддерживает эту возможность путем сопоставления информации из двух файлов метрик – для операторов и вычислителей (*.ops и *.cls соответственно, рис. 1). Имеется возможность задавать совпадение по множеству свободно назначаемых признаков для любого диапазона операторов/вычислителей. Условием выполнимости данного оператора на заданном вычислителе является соотношение $minVal\_i \le Val\_i \le maxVal\_i$ для одинакового i, где Val_i, $minVal\_i$, $maxVal\_i$ – числовые значения данного параметра для оператора и вычислителя соответственно.

Разработка расписания для выполнения программы на гетерогенном поле параллельных вычислителей является более сложной процедурой относительно ранее описанных и здесь упор делается на Lua-программирование. Так как на одном ярусе ЯПФ могут находиться операторы, требующие для выполнения различных вычислителей, полезной может служить метафора расщепления ярусов ЯПФ на семейства подъярусов, каждое из которых соответствует блоку вычислителей с определенными возможностями. Все операторы данного яруса обладают одинаковыми возможностями выполнения, последовательность обработки их в пределах яруса/подъяруса в первом приближении произвольна. На рис. 8а показано расщепление операторов на одном из ярусов ЯПФ в случае наличия 11 параллельных вычислителей трех типов, на рис. 8б – результат расчета реального плана выполнения параллельной программы на гетерогенном поле параллельных вычислителей (3 типа вычислителей по 5, 3 и 4 штук соответственно, номера исполняемых операторов скрыты).

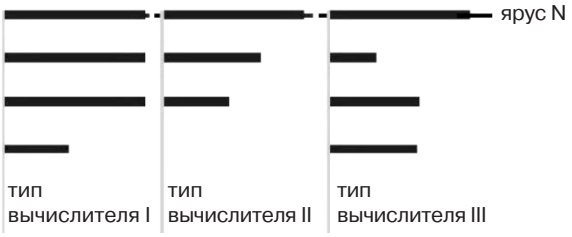
В этом случае общее время T решения задачи определяется суммой по всем ярусам максимальных значений времен выполнения операторов на подъярусах данного яруса:

$$T = \sum_j \left(\max_{k_j} \sum_i t_{ik} \right),$$

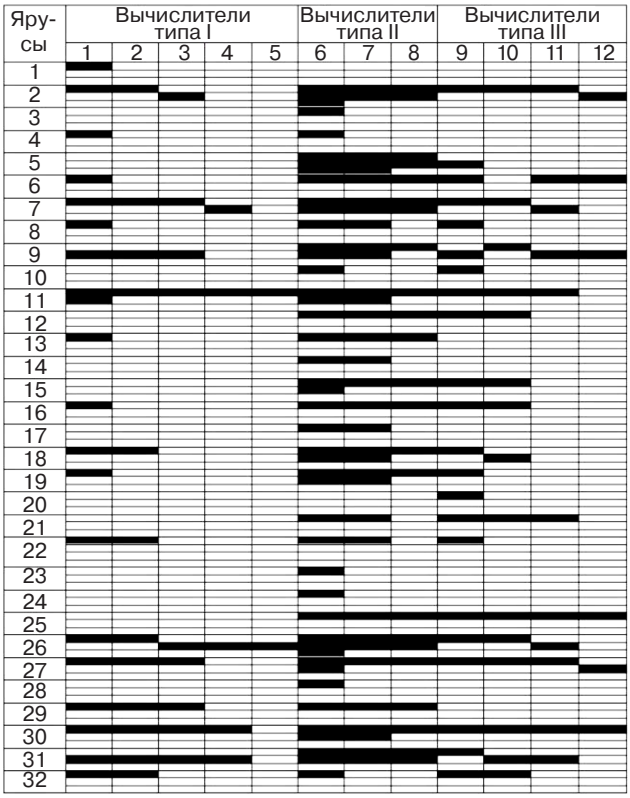
где j – число ярусов, i – число подъярусов на данном ярусе, k_j – типы вычислителей на j-м ярусе, t_{ik} – время выполнения оператора типа i на вычислителе типа k.

Если ставится задача достижения максимальной производительности, то возможно определить число вычислителей конкретного типа, минимизирующее параметр T (например, решение обратной задачи оптимизации по определению соотношения числа вычислителей разных типов). Задача минимизации общего времени решения T усложняется в случае возможности выполнения каждого оператора на нескольких вычислителях вследствие неоднозначности параметра t_{ik} в вышеприведенном выражении; здесь необходима дополнительная «балансировка» по подъярусам.

[5] Коэффициент Джини: все ли равны? «*ОТКРЫТЫЙ ЖУРНАЛ*» – медиа об инвестициях и финансах. https://journal.open-broker.ru/economy/koefficient-dzhini/. Дата обращения 31.03.2022. [Gini coefficient: Are everyone equal? *Open Journal*, an investment and finance medium. https://journal.open-broker.ru/economy/koefficient-dzhini/. Accessed March 31, 2022 (in Russ.).]



(а)



(б)

Рис. 8. Схема расщепления ярусов ЯПФ на семейства подъярусов при решении задачи определения расписания для гетерогенного поля параллельных вычислителей:
(а) схема, (б) результат расчета плана выполнения реальной параллельной программы

ОБСУЖДЕНИЕ

В ходе исследований подтвердилась возможность постепенного итерационного улучшения (в заданном направлении) эвристических сценариев преобразования исходных ЯПФ различных алгоритмов. В целом возможна разработка более быстрых (но с несколько худшим качеством по целевому назначению) и относительно медленных при более высоком качестве сценариев (фактически уровень оптимизации).

Хотя акцент данной работы ориентирован на величину вычислительной сложности сценариев получения планов (расписаний) выполнения параллельных программ, представленная программная система, по мнению автора, должна показать свою эффективность и в решении задач многомерной оптимизации при использовании известных алгоритмов (данная программная система при этом является *математической моделью объекта оптимизации*). Для реализации данного варианта использования обсуждаемая компьютерная система позволяет работать в режиме командной строки.

Несмотря на невысокую вычислительную сложность получения ЯПФ по ИГА, метод использования ЯПФ в качестве основы для построения планов выполнения параллельных программ обладает недостатком, заключающимся в невозможности простого учета времени выполнения операций. Вследствие этого время выполнения связки операторов на одном ярусе ЯПФ приходится считать равным времени выполнения самого «медленного» из них. Подход, заключающийся в целенаправленном перемещении операторов по ярусам ЯПФ, дает возможность сортировки на каждом ярусе операторов по максимально близкому времени их выполнения.

В ходе экспериментов выявилась значительная дисперсия свойств алгоритмов, представленных информационными графами, по возможности формирования планов параллельного их выполнения с максимальной плотностью кода. При этом разные алгоритмы требуют различных методов эффективного их преобразования. Представляется важной задача априорного (еще до реорганизации ЯПФ, в момент ее получения) определения сценариев для эффективного целенаправленного ее реформирования с заданной целью. Инструментом здесь должно служить создание системы классификации алгоритмов по некоторым параметрам, определяющим эффективные методы их преобразования. Возможно, перспективным может быть применение формальных методов искусственного интеллекта для решения указанной задачи.

Все проведенные эксперименты показывают, что выявленные тенденции усиливаются по мере увеличения размерности обрабатываемых данных. Это дает уверенность в сохранении полученных в результате моделирования тенденций при масштабировании по объемам обрабатываемых данных.

В соответствии с присущим эвристическому подходу итерационным принципом постепенного приближения к наилучшему решению рассматриваемой задачи имеется уверенность в возможности количественного улучшения (относительно вышеприведенных параметров) методик определения расписаний выполнения программ на заданном или определенном решением оптимизационной задачи поле параллельных вычислителей.

Важным продуктивным свойством разработанной программной системы является возможность решения обратных задач определения параметров

вычислительной системы в соответствии с заданными требованиями к самому процессу выполнения программ, например, к времени выполнения.

ЗАКЛЮЧЕНИЕ

В целом разработанный программный комплекс подтвердил эффективность в исследовании параметров скрытого параллелизма в произвольных алгоритмах и рационального использования его при обработке данных. Подход с применением скриптового языка для разработки эвристических методов (сценариев) целенаправленного преобразования форм информационного графа алгоритмов показал большую гибкость и прозрачность для исследователя. При этом гибкость достигается использованием интерпретируемого скриптового языка, а скорость обработки – возможностями исполняемого кода компилируемого языка родительского приложения.

Целевыми потребителями разработанных методов генерации расписаний параллельного выполнения программ в первую очередь являются разработчики трансляторов и виртуальных машин, исследователи свойств алгоритмов в направлении нахождения и использования потенциала скрытого их параллелизма. При практическом применении предлагаемой методики придется учесть целый ряд прикладных вопросов реализации, включая известные проблемы использования указателей, конфликтов операций работы с памятью в связках инструкций Load/Store и др., не изменяющих кардинально предложенную методику.

Разработанное программное обеспечение и методики, а именно, приемы выявления скрытого параллелизма и его параметров в произвольных алгоритмах, способы построения рациональных планов (расписаний) выполнения параллельных программ на заданном поле вычислителей, в течение нескольких лет применяются при обучении студентов в российских университетах. Это позволило повысить компетенции учащихся в области параллелизации обработки данных.

СПИСОК ЛИТЕРАТУРЫ

1. Воеводин В.В., Воеводин Вл.В. *Параллельные вычисления*. СПб.: БХВ-Петербург; 2004. 608 с.
2. Bakanov V. Research and selection of rational methods for obtaining framework of schedules for the parallel programs execution. In: Silhavy R., Silhavy P., Prokopova Z. (Eds.). *Data Science and Intelligent Systems. CoMeSySo 2021. Lecture Notes in Networks and Systems*. V. 231. Springer, Cham. https://doi.org/10.1007/978-3-030-90321-3_22
3. Федотов И.Е. *Параллельное программирование. Модели и приемы*. М.: СОЛОН-Пресс; 2018. 390 с. ISBN 978-5-91359-222-4
4. Баканов В.М. Управление динамикой вычислений в процессорах потоковой архитектуры для различных типов алгоритмов. *Программная инженерия*. 2015;9:20–24.
5. Bakanov V.M. Software complex for modeling and optimization of program implementation on parallel calculation systems. *Open Comput. Sci*. 2018;8(1): 228–234. https://doi.org/10.1515/comp-2018-0019
6. Padua D. (Ed.). *Encyclopedia of parallel computing*. NY: Springer; 2012. 2195 p. https://doi.org/10.1007/978-0-387-09766-4
7. Dennis J.B., Misunas D.P. A preliminary architecture for a basic data-flow processor. In: *Proc. Second Annual Symp. Computer Architecture* (*ISCA 75*). 1975. P. 126–132. https://doi.org/10.1145/642089.642111
8. Kukunas J. *Power and performance: Software analysis and optimization*. Morgan Kaufman, Elsevier Inc.; 2015. 300 p.
9. McNairy C., Soltis D. Itanium 2 processor microarchitecture. *IEEE Micro*. 2003;23(2):44–55. https://doi.org/10.1109/MM.2003.1196114

REFERENCES

1. Voevodin V.V., Voevodin Vl.V. *Parallel'nye vychisleniya* (*Parallel computing*). St. Petersburg: BHV-Petersburg; 2004. 608 p. (in Russ.).
2. Bakanov V. Research and selection of rational methods for obtaining framework of schedules for the parallel programs execution. In: Silhavy R., Silhavy P., Prokopova Z. (Eds.). *Data Science and Intelligent Systems. CoMeSySo 2021. Lecture Notes in Networks and Systems*. V. 231. Springer, Cham. https://doi.org/10.1007/978-3-030-90321-3_22
3. Fedotov I.E. *Parallel'noe programmirovanie. Modeli i priemy* (*Parallel programming. Models and techniques*). Moscow: SOLON-Press; 2018. 390 p. (in Russ.). ISBN 978-5-91359-222-4
4. Bakanov V.M. Dynamics control computing in the processors data flow architecture for different types of algorithms. *Programmnaya inzheneriya = Software Engineering*. 2015;9:20–24 (in Russ.).
5. Bakanov V.M. Software complex for modeling and optimization of program implementation on parallel calculation systems. *Open Comput. Sci*. 2018;8(1): 228–234. https://doi.org/10.1515/comp-2018-0019
6. Padua D. (Ed.). *Encyclopedia of parallel computing*. NY: Springer; 2012. 2195 p. https://doi.org/10.1007/978-0-387-09766-4
7. Dennis J.B., Misunas D.P. A preliminary architecture for a basic data-flow processor. In: *Proc. Second Annual Symp. Computer Architecture* (*ISCA 75*). 1975. P. 126–132. https://doi.org/10.1145/642089.642111
8. Kukunas J. *Power and performance: Software analysis and optimization*. Morgan Kaufman, Elsevier Inc.; 2015. 300 p.
9. McNairy C., Soltis D. Itanium 2 processor microarchitecture. *IEEE Micro*. 2003;23(2):44–55. https://doi.org/10.1109/MM.2003.1196114

10. Таненбаум Э., Остин Т. *Архитектура компьютера*: пер с англ. СПб.: Питер; 2019. 816 с. ISBN 978-5-4461-1103-9
11. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. *Introduction to algorithms*. 3rd ed. London: MIT Press; 2009. 1292 p.
12. McConnell J.J. *Analysis of algorithms: An active learning approach*. Jones & Bartlett Publishers; 2008. 451 p.
13. Гэри М., Джонсон Д. *Вычислительные машины и труднорешаемые задачи*: пер. с англ. М.: Книга по требованию; 2012. 420 с. ISBN 978-5-458-26100-5
14. Ierusalimschy R. *Programming in Lua*. 3rd ed. PUC-Rio, Brasil, Rio de Janeiro; 2013. 348 p.
15. Fisher J.A. Very long instruction word architectures and the ELI-512. In: *Proceedings of the 10th Annual International Symposium on Computer Architecture* (*ISCA '83*). 1983. P. 140–150. https://doi.org/10.1145/800046.801649
16. Babb R.I. Parallel processing with large-grain data flow techniques. *Computer*. 1984;17(7):55–61. https://doi.org/10.1109/MC.1984.1659186

10. Tanenbaum E., Ostin T. *Arkhitektura komp'yutera* (*Computer architecture*). Transl. from Eng. St. Petersburg: Piter; 2019. 816 p. (in Russ.). ISBN 978-5-4461-1103-9
11. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. *Introduction to algorithms*. 3rd ed. London: MIT Press; 2009. 1292 p.
12. McConnell J.J. *Analysis of algorithms: An active learning approach*. Jones & Bartlett Publishers; 2008. 451 p.
13. Garey M., Johnson D. *Vychislitel'nye mashiny i trudnoreshaemye zadachi* (*Computing machines and difficult tasks*). Transl. from Eng. Moscow: Kniga po trebovaniyu; 2012. 420 p. (in Russ.). ISBN 978-5-458-26100-5
[Garey M., Johnson D. *Computers and intractability*. San Francisco; 1979. 338 p.]
14. Ierusalimschy R. *Programming in Lua*. 3rd ed. PUC-Rio, Brasil, Rio de Janeiro; 2013. 348 p.
15. Fisher J.A. Very long instruction word architectures and the ELI-512. In: *Proceedings of the 10th Annual International Symposium on Computer Architecture* (*ISCA '83*). 1983. P. 140–150. https://doi.org/10.1145/800046.801649
16. Babb R.I. Parallel processing with large-grain data flow techniques. *Computer*. 1984;17(7):55–61. https://doi.org/10.1109/MC.1984.1659186

Об авторе

Баканов Валерий Михайлович, д.т.н., профессор, профессор кафедры КБ-5 «Аппаратное, программное и математическое обеспечение вычислительных систем» Института кибербезопасности и цифровых технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: e881e@mail.ru. Scopus Author ID 6505511355, ResearcherID L-1314-2015, SPIN-код РИНЦ 8868-4594, https://orcid.org/0000-0002-1650-038X

About the author

Valery M. Bakanov, Dr. Sci. (Eng.), Professor, Department of Hardware, Software and Mathematical Support of Computing Systems, Institute of Cybersecurity and Digital Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: e881e@mail.ru. Scopus Author ID 6505511355, ResearcherID L-1314-2015, RSCI SPIN-code 8868-4594, https://orcid.org/0000-0002-1650-038X