УДК 004.4'2 + 004.51 DOI: 10.17587/prin.8.58-65

В. М. Баканов, д-р техн. наук, проф., e-mail: vbakanov@hse.ru, Национальный исследовательский университет "Высшая школа экономики" (НИУ ВШЭ), Москва

Программный комплекс для разработки методов построения оптимального каркаса параллельных программ

Проанализированы подходы к разработке рациональных (стремящихся к оптимальным) методов построения планов выполнения прикладных вычислительных задач на реальных параллельных вычислительных системах (каркасов выполнения), а также инструментальная программная среда для реализации таких методов. Предложены критерии и параметры оптимизации методов планирования. Представлены результаты применения некоторых разработанных стратегий построения рациональных планов выполнения параллельных программ.

Ключевые слова: *графовые представления алгоритма, анализ информационной структуры программы, ярусно-параллельная форма информационного графа, рациональные параметры выполнения параллельных программ, стратегия построения рационального плана выполнения параллельной программы*

Введение

В настоящее время и в ближайшем будущем одним из реальных путей повышения производительности вычислительных систем является подход на основе распараллеливания процессов обработки данных (далее — параллелизация). Путь повышения тактовой частоты процессов ограничивается фундаментальными законами и технологией их производства.

Вместе с тем параллелизация (возможность одновременного, параллельного выполнения различных частей одной программы на независимо работающих вычислителях) требует от специалистов по алгоритмике и программистов дополнительных усилий по выявлению в алгоритме участков, которые могут быть исполнены параллельно (один из вариантов декомпозиции). Основное требование к таким блокам (*зернам* или *гранулам*) параллелизма состоит в их независимости (ортогональности) по данным.

Обзор исследований в данной области

Анализ алгоритмов на наличие потенциала параллелизма, часто именуемый *исследованием тонкой информационной структуры алгоритма*, непрост. Поэтому говорят о скрытом (не фиксируемом при *поверхностном рассмотрении*) параллелизме. Конструктивным инструментарием при таком анализе является представление программ графовыми моделями. В нашей стране разработкой методов анализа структуры алгоритмов на основе графовых моделей занимались В. В. Воеводин и Вл. В. Воеводин [1].

Ситуация с использованием параллелизма усложняется необходимостью рационального применения ресурсов конкретной многопроцессорной вычислительной системы (МВС). Эти ресурсы (число и возможности параллельно работающих вычислителей) в каждом отдельном случае ограничены, поэтому задача планирования использования ресурсов является важной. Несмотря на наличие архитектур параллельных вычислительных систем, не нуждающихся в априорном планировании (например, потоковый вычислитель аппаратно определяет очередность выполнения операторов на этапе выполнения программ), именно априорному планированию при реализации программ на МВС в настоящее время отдается приоритет.

Априорный (до этапа выполнения) анализ структуры алгоритмов не является самодостаточным, он лишь основа для оценки эффективности выполнения программы на конкретной МВС. Учет конкретики МВС при этом во многих случаях невозможен аналитически и требует моделирования и оптимизации процесса. С этих позиций направление разработки инструментальных программных систем для решения задач эффективного использования МВС актуально.

Как правило, для представления и анализа программ используют (наряду с иными формами) информационные графы (например, проект AlgoWiki, [2]). При таком подходе для максимального повышения быстродействия оптимизации подвергают критические участки программы, не имеющие ветвлений. В этом случае информационный граф зачастую достаточен для описания алгоритма. Целью AlgoWiki является "дать исчерпывающее описание алгоритма, которое поможет оценить его потенциал применительно к конкретной параллельной вычислительной

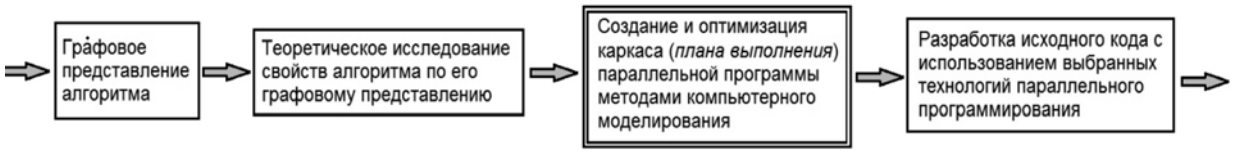


Рис. 1. Фрагмент цикла разработки эффективной параллельной программы

платформе" [2]. Следующим (перед процедурой собственно кодирования) и естественным этапом в цикле разработки параллельного приложения должно являться моделирование выполнения программы на конкретной МВС. При этом формируется оптимальный каркас (фактически план выполнения) параллельной программы (рис. 1).

При представлении программы информационным графом алгоритма (ИГА, вычислительная модель "операторы — операнды") вершины графа соответствуют преобразователям информации (операторам), а дуги — информационным связям операторов (операндам). В данном случае термин *операторы* достаточно условен, ибо под *операторами* понимают отдельные последовательности вычислений (блоки, подзадачи) любого размера в зависимости от соглашений, принятых при декомпозиции исходной программы. К свойствам ИГА относят: ацикличность, однонаправленность, детерминированность. Дуги ИГА нагружены значениями объема пересылаемых данных, вершины — значениями вычислительной сложности операторов, выраженными в машинных командах или тактах процессора. Эти данные не относятся к каркасу и конкретизируются на следующем этапе разработки параллельной программы (выбор технологии параллельного программирования и собственно кодирование) для оценки времени ее выполнения. Время пересылки данных определяется технологией обмена с учетом латентности коммуникационной среды, а время выполнения операторов — параметрами процессора.

Удобным и полезным методом выявления параллелизма является представление ИГА в ярусно-параллельной форме (ЯПФ). При таком представлении операторы, обладающие способностью выполняться независимо друг от друга (фактически параллельно), располагаются на одном уровне (*ярусе*). Формально каждый ярус ЯПФ является одним из сечений ИГА, удовлетворяющих условию независимости по данным, находящимся на этом ярусе операторов. Трудоемкость построения ЯПФ квадратичная от числа операторов.

Глубина ЯПФ, т. е. число ярусов, определяет общее время выполнения алгоритма, ширина ЯПФ — максимальное число задействованных отдельных (параллельно работающих) вычислителей, например, отдельных ядер процессора, данной МВС. В случае начала построения ЯПФ с исходных данных получаем "верхний" вариант ЯПФ (операторы располагаются на наиболее близких к начальным ярусах). При построении начиная с выходных данных получаем "нижний" вариант. Два таких представления ограничивают весь диапазон возможного расположения операторов на ярусах ЯПФ. Обычно считается, что в рамках каждого яруса операторы выполняются синхронно, а поле параллельных вычислителей — гомогенно. На рис. 2 показана ЯПФ графа простой процедуры вычисления вещественных корней полного квадратного уравнения $ax^2 + bx + c = 0$ (ЯПФ в *каноническом* виде, вследствие простоты алгоритма соответствие вершин графа вычислительным операциям не приводится).

Формально каноническая ЯПФ уже представляет собой некоторый каркас (план выполнения) отдельных частей программы — номера ярусов определяют последовательность выполнения параллельных групп операторов на данном ярусе. Однако подобное расписание реализуемо лишь на гипотетической МВС с бесконечно большим числом параллельных вычислителей (согласно *концепции неограниченного*

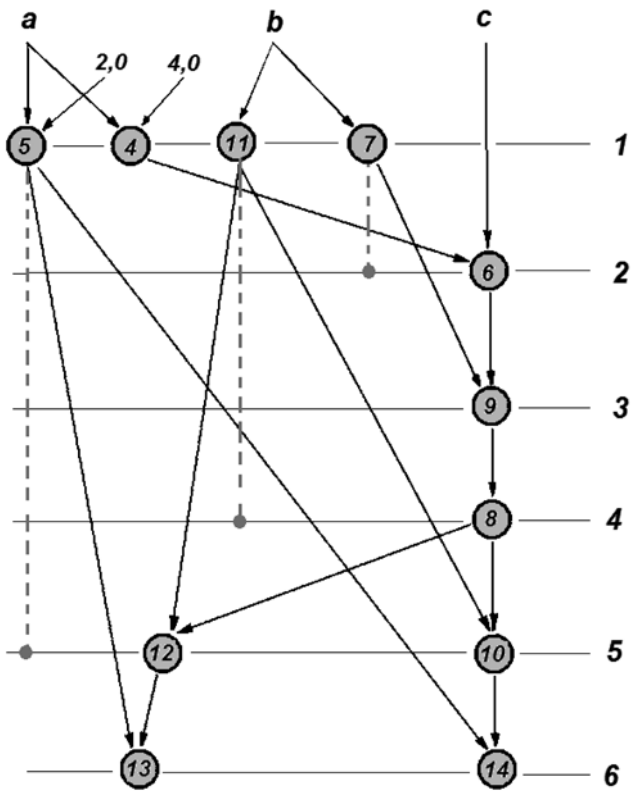


Рис. 2. Ярусно-параллельная форма процедуры вычисления вещественных корней полного квадратного уравнения (в "верхнем" варианте, крайние справа цифры — номера ярусов ЯПФ; штриховыми линиями показан диапазон возможного расположения операторов по ярусам)

параллелизма). Реальные ЯПФ информационных графов обладают большой неравномерностью в распределении числа операторов по ярусам (показательный пример — применяемый для ассоциативных операций *граф сдваивания*). Использование ресурсов МВС в таком случае очень нерационально — с точки зрения выполнения каждой программы максимум эффективности использования ресурсов обычно достигается при равномерном или близком к равномерному распределении операторов по ярусам. Вследствие стремления разработчиков к быстрейшему выполнению вычислений обычно останавливаются на построении статичного плана исполнения параллельной программы (миграция подзадач в пределах конкретного задания между вычислителями резко снижает производительность).

Применяемые при исследовании методы

Практически в любом ЯПФ имеется вариативность в расположении вершин графа (операторов) по ярусам (перемещение операторов между ярусами ограничивается соблюдением информационных зависимостей); для ЯПФ графа на рис. 2 оператор *5* может быть перемещен "вниз" на любой ярус со второго по пятый, оператор *11* — на любой ярус со второго по четвертый, оператор 7 — на второй ярус. Эта особенность ЯПФ дает возможность оптимизации ЯПФ (в смысле, например, достижения наибольшей равномерности распределения операторов по ярусам). В представленном на рис. 2 случае возможна "разгрузка" первого яруса от операторов, например, *5* и *11*; в результате программа может быть выполнена на двух (вместо четырех) параллельно работающих вычислителях за то же время.

Данная задача близка к проблематике разбиения графов и, согласно работе [3], относится к классу *NP*-полных. Для ИГА большого размера трудоемкость оптимизации путем перебора запредельно велика (задача перестановок с учетом влияния изменения конфигурации ЯПФ после каждой), существующие алгоритмы разбиения графов являются эвристическими; с учетом этого корректнее говорить о нахождении не оптимального, а приближающегося к оптимальному *рационального* решения. Параметрами оптимизации служит все множество планов выполнения операторов параллельной программы (с учетом сохранения информационных связей). В качестве критерия обычно принимают максимум скорости выполнения и/или полноту использования существующего оборудования (многокритериальная оптимизация). Для практики необходимо иметь гибкий аппарат для решения задачи планирования с разной степенью оптимизации (вследствие *NP*-полноты задачи понимая эту процедуру как последовательное повышение качества эвристик в сторону улучшения критерия оптимизации) — от "быстрых" методов, основанных, например, на использовании ЯПФ, до более трудоемких (методы "ветвей и границ", эволюционные и др.) для получения лучших решений. С этой точки зрения ценностью является получение количественных значений текущих приближений в качестве отправной точки для проведения дальнейших шагов оптимизации.

К ограничениям МВС относят: число параллельно работающих вычислителей, объемы их памяти, функциональные возможности, параметры коммуникационной среды. Выполнение программы с учетом этих ограничений достигается эквивалентными (не изменяющими информационных зависимостей в графе) преобразованиями ЯПФ-разбиений ИГА.

Будем использовать следующие параметры ЯПФ информационного графа:

- H — глубина ЯПФ графа, определяется числом ярусов в нем;
- W_i — ширина i-го яруса (число операторов на нем);
- W — ширина ЯПФ графа, определяется максимумом операторов по всем ярусам, $W = \max(W_i)$;
- $\bar{W}$ — средняя ширина ЯПФ; $\bar{W} = \frac{1}{N}\sum_{i=1}^{i=N} W_i$;

N — число ярусов ЯПФ;

- k — степень (коэффициент) неравномерности распределения операторов по ярусам ЯПФ графа; $k = \max(W_i) / \min(W_i)$;
- σ — среднее квадратичное отклонение ширин ярусов; $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{i=N}\left(W_i - \bar{W}\right)^2}$.

Сформулируем и перечислим далее типовые постановки оптимизационной задачи.

◇ Максимально полная загрузка неизвестного заранее (но не меньшего $\bar{W}$) числа параллельно работающих вычислителей имеющейся МВС при минимальном времени выполнения задачи: $k \to 1{,}0$ при $H = H_0 = \text{const}$, где H_0 — длина критического пути, а именно минимальная из всех возможных высот ЯПФ данного графа.

◇ Наиболее полное использование доступных ресурсов (вычислительных узлов) имеющейся МВС при допущении увеличения времени выполнения (отход от канонической ЯПФ): $\max(W_i) \leqslant W_0$ при $H \geqslant H_0$, здесь W_0 — число доступных параллельно работающих вычислителей в заданной МВС; однако желательно $H - H_0 \to \min$ (минимизация времени выполнения). Для наглядности полезно представить этот процесс как подготовку связки, состоящей из не более чем заданного числа W_0 заведомо независимых (ортогональных) по операндам команд для вычислительной системы в аппаратно-программной идеологеме микропроцессорной архитектуры с явным параллелизмом команд (EPIC, *Explicitly Parallel Instruction Computing*) для параллельных вычислителей архитектуры сверхдлинного командного слова (VLIW, *Very Long Instruction Word*).

Для реализации и оптимизации эвристических алгоритмов поиска рациональных (стремящихся к оптимальным) планов выполнения параллельных частей параллельных программ разработано

специализированное программное обеспечение инструментального уровня, в котором для описания стратегий решения задачи использован встроенный скриптовый язык программирования Lua [4]. Данный встроенный язык программирования выбран по причинам его компактности (при его применении возрастание размера исполняемого файла родительского приложения не превышает ≈100 Кбайт) и простоты обучаемости для его использования (в основном Lua следует синтаксису языка C), широким функциональным возможностям (поддерживается объектно-ориентированная модель программирования и др.), бесплатности при наличии исходных текстов. Перечисленные обстоятельства подтверждают правильность выбора Lua в качестве управляющего языка для создания представлений ИГА. В целом возможностей Lua вполне достаточно для реализации наиболее трудоемких методов оптимизации задачи планирования. Клиентская часть программного инструментария (предварительное информационное сообщение приведено в работе [5]) разработана на языке программирования C++ и является 32-битным GUI-приложением Windows (рис. 3).

Набор API-вызовов разработанной системы позволяет реализовывать любой из возможных критериев оптимизации, включая их комбинации, так как выбор критерия осуществляет собственно пользователь, на основе задач, им поставленных. Частично пересекающимися по функциональным возможностям с данным проектом являются проекты V-Ray и ПАРУС (оба — разработки МГУ им. М. В. Ломоносова). Известны распространяемые по лицензии GNU с открытым кодом системы METIS и ParMETIS (университет Миннесоты, 1998—2003 гг.), функциональные возможности которых для данного случая явно избыточны — в реальных программах эффективно распараллеливать имеет смысл лишь отдельные участки кода, критичные по времени выполнения или по использованию аппаратных ресурсов. Следует отметить, что существует и ряд других пакетов, реализующих процедуры разбиения графов для последовательных и параллельных вычислителей.

Пакеты с открытым кодом обычно требуют пересборки после настройки под конкретную задачу путем изменения исходного текста. В отличие от других предлагаемая программная система представлена в виде скомпилированных исполняемых файлов и пересборки не требует. Гибкость достигается достаточным набором API-функций и, при желании, пользователь может совершать все манипуляции с ИГА средствами самого встроенного языка программирования, используя API-вызовы, например, вполне возможна работа и без явного представления ИГА в ЯПФ. Благодаря перечисленным качествам предлагаемая система (по сравнению с известными) рассчитана на более широкий круг специалистов в области исследования и оптимизации параллельного программирования.

Программный интерфейс рассматриваемой системы включает три типа API-вызовов (всего их около 30 шт., причем каждый является Lua-"оберткой" соответствующей C-функции):

- информационные вызовы, которые служат для получения информации об ИГА и его ЯПФ; именно на основании этих данных в дальнейшем принимается решение о применимости одной из стратегий обработки ИГА для решения поставленной задачи, реализуемой в дальнейшем на встроенном языке программирования;
- акционные вызовы, которые служат для реализации конкретных стратегий решения задачи построения расписания;
- вспомогательные вызовы, предназначенные для вывода рассчитанных данных в текстовом и графическом видах для обмена данными с иными приложениями, для взаимодействия с файловой системой и т. п.

Примерами информационных вызовов являются: определить общее число вершин и дуг ИГА, смежную вершину по заданной, общее число ярусов ЯПФ и число операторов на заданном ярусе, диапазон ярусов допустимого расположения каждого оператора, характер распределения ширин ярусов ЯПФ, число и параметры параллельных вычислителей и др. К числу акционных вызовов относятся: создать ЯПФ по указанному ИГА (реализовано в виде отдельного API-вызова); переместить заданный оператор на конкретный ярус ЯПФ (с учетом ненарушения информационных зависимостей), создать новый пустой ярус; уничтожить пустой ярус и др. Выполнение API-функций подробно протоколируется в файле с уникальным именем.

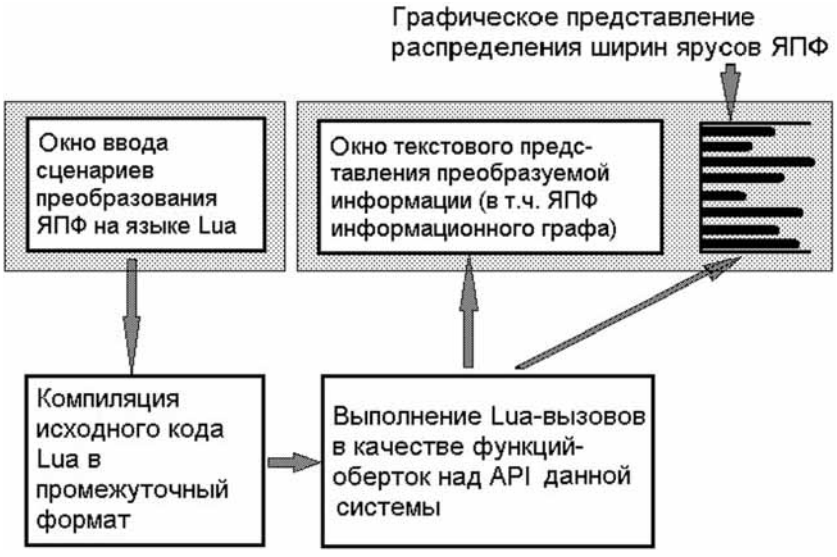


Рис. 3. Схема взаимодействия компонентов разработанной инструментальной программной системы и принципы функционирования пользовательского интерфейса

Данные расчетов выдаются как в текстовом, так и в графическом (ленточная диаграмма распределения операторов по ярусам ЯПФ) видах (см. рис. 3, 4) и могут быть сохранены в распространенных форматах.

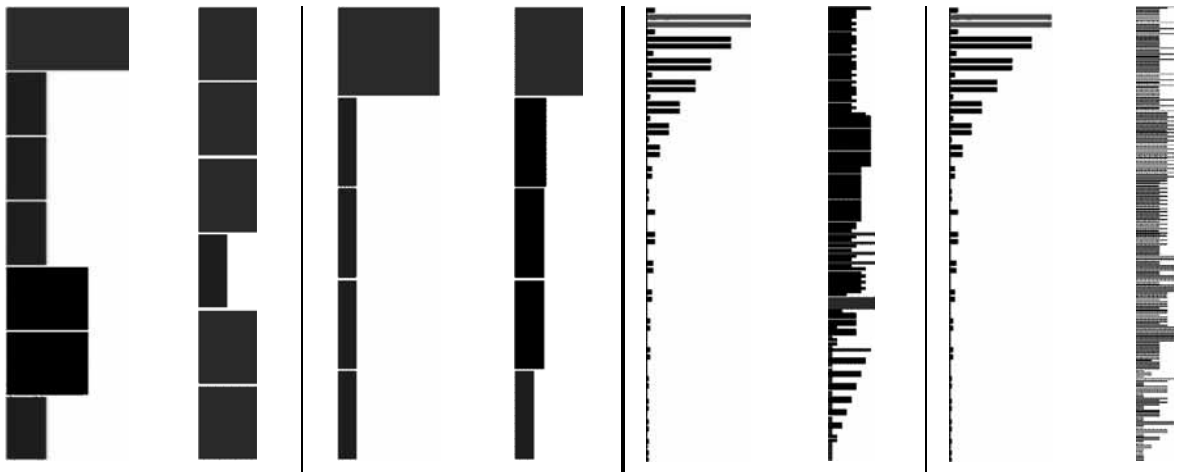
В случае применения данной системы в качестве компонента распараллеливающего компилятора, граф алгоритма априорно существует (компилятор при работе проводит анализ программы на информационные зависимости, т. е. фактически строит граф). Автор для получения корректного ИГА в течение ряда лет использует программный симулятор вычислителя потоковой (DATA-FLOW) архитектуры [6], в котором программа отлаживается и в дальнейшем экспортируется в файловый формат списка смежных вершин. Особенностью этого симулятора является возможность моделирования асинхронного режима выполнения операторов (для каждого оператора задается время исполнения в условных единицах).

Результаты, их обсуждение и перспективы

Одной из типовых является задача определения эффективных стратегий балансировки ЯПФ информационного графа алгоритма, здесь под балансировкой понимается равномерность распределения операторов по ярусам ЯПФ. На первой стадии из всех реалий конкретной МВС учитывается только число параллельных вычислителей. Разработанные методы учета гетерогенности (по параметрам объема памяти, вычислениям с различными типами данных, векторных вычислений и т. п.) вычислительного поля с возможностью выбора выполнения определенного оператора на соответствующем по параметрам вычислителе потребовали дополнения системы приблизительно 20 API-вызовами.

В качестве *параметра эффективности* собственно стратегии преобразования ЯПФ использовано число перестановок операторов с яруса на ярус (аналогично задачам сортировки) для получения заданного результата (лучшей стратегии соответствует минимум числа перестановок). При таком подходе превышение высоты ЯПФ над значением критического пути удобно рассматривать в качестве штрафной функции.

На рис. 4 в графической интерпретации (ленточные диаграммы распределения числа операторов по ярусам) приведены результаты применения различ-



Исходная ЯПФ	Преобразованная ЯПФ	Исходная ЯПФ	Преобразованная ЯПФ	Исходная ЯПФ	Преобразованная ЯПФ	Исходная ЯПФ	Преобразованная ЯПФ
(H=6; W=4; k=4; σ =1,07)	(H=6; W=2; k=2; σ =0,373)	(H=5; W=125; k=5; σ =40)	(H=5; W=85; k=3,8; δ=25,3)	(H=63; W=90; k=90; σ =21,8; $\overline{W}$=12,8)	(H=141; W=10; k=10; δ=2,69; $\overline{W}$=5,71)	(H=63; W=90; k=90; σ =21,8; $\overline{W}$=12,8)	(H=230; W=5; k=5; σ =1,15; $\overline{W}$=3,50)
Вычисление вещественных корней полного квадратного уравнения ($\overline{W}$ =1,83)		Умножение квадратных матриц пятого порядка классическим методом ($\overline{W}$ =45)		Решение системы линейных уравнений (СЛАУ) десятого порядка методом Гаусса (W_0=10)		Решение СЛАУ десятого порядка методом Гаусса (W_0=5)	
Преобразование ЯПФ согласно стратегии #01 (условие H=const)				Преобразование ЯПФ согласно стратегии #02 (условие $W<=W_0$)			

Рис. 4. Диаграммы распределения операторов по ярусам для исходных и преобразованных ЯПФ с помощью различных стратегий преобразования для разных алгоритмов (вариант гомогенного поля параллельных вычислителей)

Таблица 1

Эффективность применения стратегии балансировки ширины ЯПФ при неизменной высоте (числитель — для исходной ЯПФ, знаменатель — для преобразованной ЯПФ)

ИГА (дуг/операторов)	Число ярусов ЯПФ	Средняя ширина ЯПФ	Неравномерность ширины ЯПФ	Среднее квадратичное отклонение ширины ЯПФ	Число перестановок операторов
slau_2a (18/9)	7	1,29	2/2	0,452/0,452	0
slau_3a (56/28)	13	2,15	6/6	1,70/1,51	4
slau_5a (230/115)	25	4,60	20/20	5,42/5,11	13
slau_10a (1610/805)	63	12,8	90/90	21,8/20,7	81
mnk_10 (132/66)	16	4,13	22/22	4,79/4,78	1
mnk_20 (252/126)	26	4,85	42/42	7,52/7,51	1
korr_10 (174/88)	15	5,87	32/32	7,15/7,14	4
korr_20 (334/168)	25	6,72	62/44	11,4/7,77	22
m_matr_5 (450/225)	5	45,0	5/3,40	40/20,7	30
m_matr_10 (3800/1300)	10	190	10/6	270/137	407

ных стратегий (обе относятся к классу "быстрых" методов) преобразования ЯПФ для двух случаев. Первый из них — стратегия #01 — перемещение операторов с наиболее нагруженных на наименее нагруженные ярусы при условии сохранения высоты ЯПФ, критерий остановки алгоритма — перебор всех операторов, которые могут быть перенесены "с холмов во впадины" в целях балансировки ЯПФ. Случай второй — стратегия #02 — балансировка при условии увеличения высоты ЯПФ, критерий останова — "ужатие", т. е. снижение ширины ЯПФ до значения, не превышающего заданного, для нескольких ИГА распространенных вычислительных процедур.

Эффективность обработки данных иллюстрируется данными табл. 1 (стратегия #01) и табл. 2 (стратегия #02 при ограничениях ширины ЯПФ разного размера). Информационные графы slau_2a, slau_3a, slau_5a и slau_10a — процедуры решения СЛАУ порядков 2, 3, 5, 10, соответственно, классическим методом Гаусса; mnk_10 и mnk_20 — процедуры линейной аппроксимации по методу наименьших квадратов для 10 и 20 точек; korr_10 и korr_20 — процедуры

Таблица 2

Эффективность применения стратегии получения ЯПФ не более заданной ширины совместно с балансировкой ширин ярусов (для ширины ЯПФ не более 5)

ИГА (дуг/операторов)	Число ярусов ЯПФ	Средняя ширина ЯПФ	Неравномерность ширины ЯПФ	Среднее квадратичное отклонение ширины ЯПФ	Число перестановок операторов
slau_2a (18/9)	7/7	1,29/1,29	2/2	0,452/0,452	0
slau_3a (56/28)	13/15	2,15/1,87	6/3	1,7/0,806	6
slau_5a (230/115)	25/39	4,6/2,95	20/5	5,42/1,36	70
slau_10a (1610/805)	63/230	12,8/3,5	90/5	21,8/1,15	1234
mnk_10 (132/66)	16/21	4,13/3,14	22/5	4,79/1,28	27
mnk_20 (252/126)	26/35	4,85/3,6	42/5	7,52/1,18	67
korr_10 (174/88)	15/23	5,87/3,83	32/5	7,15/1,2	51
korr_20 (334/168)	25/41	6,72/4,1	62/5	11,4/1,06	124
m_matr_5 (450/225)	5/64	45/3,52	5/1,33	40/0,5	451
m_matr_10 (3800/1300)	10/544	190/3,49	10/1,33	270/0,5	6152

Таблица 3

Зависимость трудоемкости преобразований ЯПФ (в единицах числа перестановок операторов с яруса на ярус) стратегии #02 получения ЯПФ не более заданной ширины W_0

ИГА (дуг/операторов/ярусов)	Ширина W_0					
	10	8	6	4	2	1
slau_10a (1610/805/63)	926	1038	1135	1340	1601	1920
m_matr_10 (3800/1300/10)	5232	5232	5828	6152	6964	7776
korr_20 (334/168/25)	91	91	121	160	212	283
mnk_20 (252/126/26)	51	61	61	79	127	183
e313_o211_t33 (313/211/33)	52	60	103	136	207	288
e451_o276_t31 (451/276/31)	85	98	177	207	312	425
e2367_o1402_t138 (2367/1402/138)	496	613	969	1181	1730	2334
e17039_o9858_t200 (17039/9858/200)	14 579	16 129	17 418	20 250	23 521	27 507

определения коэффициента парной корреляции для 10 и 20 точек; m_matr_5 и m_matr_10 — процедуры умножения квадратных матриц порядков 5 и 10 классическим методом (гомогенное поле параллельных вычислителей). Неравномерность распределения числа операторов по ярусам ЯПФ оценивали неравномерностью распределения операторов по ярусам k, дополнительно определяли среднее квадратичное отклонение ширин ярусов.

Следует заметить, что потенциал балансировки ЯПФ при неизменной его высоте ограничен (см. табл. 1), поэтому больший практический интерес представляют стратегии с увеличением высоты ЯПФ (см. табл. 2).

Табл. 3 иллюстрирует зависимость трудоемкости преобразований ЯПФ (в единицах перестановок операторов с яруса на ярус) от заданного предела ширины ЯПФ для графов ранее рассмотренных распространенных алгоритмов и нескольких графов, сгенерированных программой-графогенератором (для данной стратегии преобразования ЯПФ рост трудоемкости близок к полиномиальной функции от степени "ужатия" ширины ЯПФ).

Как видно из данных табл. 1—3, ранее описанные (не являющиеся излишне изощренными) стратегии позволяют снизить неравномерность ширин ЯПФ при заданных ограничениях, однако с разной эффективностью для различных ИГА. В целом наблюдается повышение эффективности стратегии при увеличении сложности (и, соответственно, вариативности) ИГА. Особенно интересна и практически важна задача априорного (до проведения собственно преобразований ЯПФ) предсказания уровня эффективности стратегий (задача распознавания ситуации). Большего эффекта можно добиться путем совершенствования стратегий и разумного их совместного применения.

Заключение

Использование встроенного высокоуровневого языка программирования для моделирования методов планирования задач для параллельных вычислительных систем позволило добиться эффективности и гибкости в реализации поставленной цели, что было бы затруднительно при использовании иных подходов. Получены количественные параметры текущего приближения при совершенствовании стратегий оптимизации выполнения параллельных программ. Результаты исследований применимы для анализа алгоритмов (в плане выявления эффективности выполнения параллельных программ), при разработке распараллеливающих компиляторов и в процессе обучения специалистов в области технологий параллельного программирования.

Список литературы

1. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
2. **AlgoWiki.** Открытая энциклопедия свойств алгоритмов. URL: http://algowiki-project.org (дата обращения 01.09.2016).
3. **Гери М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
4. **Иерузалимски Р.** Программирование на языке Lua. М.: ДМК Пресс, 2014. 382 с.
5. **Баканов В. М.** Программный инструментарий анализа информационной структуры алгоритмов по их информационным графам // Параллельные вычислительные технологии (ПаВТ'2016): Тр. междунар. науч. конф. Архангельск, 28 марта — 01 апреля 2016 г. Челябинск: Издательский центр ЮУрГУ, 2016. С. 432—441.
6. **Баканов В. М.** Управление динамикой вычислений в процессорах потоковой архитектуры для различных типов алгоритмов // Программная инженерия. 2015. № 9. С. 20—24.

Program Complex for Development Methods Construction of Optimal Frame Parallel Program

V. M. Bakanov, vbakanov@hse.ru, Higher School of Economics, National Research University, 101000, Moscow, Russian Federation

Corresponding author:

Bakanov Valery M., Professor, Higher School of Economics, National Research University, 101000, Moscow, Russian Federation
E-mail: vbakanov@hse.ru

Received on October 10, 2016
Accepted on October 28, 2016

Approaches to the problem of the development of rational (seeking optimal) planning techniques (drawing up the framework, the skeleton program in terms of parallelism) to perform the tasks of parallel computing systems with homogeneous and heterogeneous field calculators. The informational graphs are used as a formalization of the algorithm. The possible criteria and parameters of the optimization planning methods with the use of stacked parallel form (SPF) information graph algorithm, or without it are described. Two basic strategies are formulated — preserving run time (without increasing the height of the original SPF) or an increase in the height of SPF. The first one is implemented on the basis of statements on location variability tiers SPF (while retaining the same information dependencies as in the original graph), the second one — by adding tiers in SPF and the transfer of the operators "top-down" on the written stage. The data on the developed for the implementation of such methods software system (the program stand) are provided. To implement methods (strategies) of rational development of parallel programs, the built-in high-level Lua scripting language is used. The confirmation of its effectiviness in this capacity is provided. The information on the set of API calls of the system is provided. A criterion of computational complexity of parallel programs execution plan building procedures is suggested. We give qualitative (in the form of strip-chart) and quantitative results of the application of some of the proposed strategies for building a rational planning of parallel programs with respect to the common algorithms of data processing.

Keywords: *graph representations of the algorithm, analysis of the information structure of the program, longline-parallel form information graph, rational parameters of parallel programs, integrated lua scripting language, strategy of building a rational plan for the parallel program execution*

For citation:

Bakanov V. M. Program Complex for Development Methods Construction of Optimal Frame Parallel Program, *Programmnaya Ingeneria,* 2017, vol. 8, no. 2, pp. 58—65.

DOI: 10.17587/prin.8.58-65

References

1. **Voevodin V. V., Voevodin Vl. V.** *Parallel'nye vychislenija* (Parallel computing), St. Petersburg, BHV-Petersburg, 2002, 608 p. (in Russian).
2. **AlgoWiki.** Open Encyclopedia properties of algorithms, available at: http: // algowiki-project.org (reference date 01.09.2016).
3. **Gary M., Johnson D.** *Vychislitel'nye mashiny i trudnoreshaemye zadachi* (Computers and Intractability), Moscow, Mir, 1982, 416 p. (in Russian).
4. **Ierusalimschy R**. *Programmirovanie na jazyke Lua* (Programming in Lua), Moscow, DMK Press, 2014, 382 p. (in Russian).
5. **Bakanov V. M.** Programmnyj instrumentarij analiza informacionnoj struktury algoritmov po ih informacionnym grafam (Software tool analyzes information structure algorithms for their information graphs), *Parallel Computing Technologies (PaVT'2016): Proceedings of the International Scientific Conference,* Arkhangelsk, 28.03—01.04.2016, Chelyabinsk, Publishing Center South Ural State University, 2016, pp. 432—441 (in Russian).
6. **Bakanov V. M.** Upravlenie dinamikoj vychislenij v processorah potokovoj arhitektury dlja razlichnyh tipov algoritmov (Computing the dynamics of management processors streaming architecture for the different types of algorithms), *Programmnaya Ingeneria,* 2015, no. 9, pp. 20—24 (in Russian).