УДК 004.4'2 + 004.51

ПРОГРАММНАЯ СИСТЕМА ДЛЯ РАЗРАБОТКИ СТРАТЕГИЙ ПОСТРОЕНИЯ ОПТИМАЛЬНОГО КАРКАСА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

Баканов В.М., д.т.н., профессор

Национальный исследовательский университет "Высшая школа экономики" (НИУ ВШЭ), Россия, 101000, г.Москва, ул.Мясницкая, д.20

Физико-технологический институт Московского технологического университета, Россия, Москва, 107996, Стромынка, 20.

@ Для переписки: Баканов В.М., e-mail: e881e@mail.ru

Работа посвящена решению проблемы создания рациональных методов разработки каркаса (плана, расписания выполнения) параллельных программ для реальных параллельных вычислительных систем. Для решения этой задачи разработана программная среда, позволяющая реализовывать различные стратегии построения каркаса выполнения параллельных программ и оценивать качество этих стратегий. Для моделирования и оптимизации методов используется встроенный скриптовый язык программирования Lua. Приводятся результаты применения некоторых предложенных 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.

PROGRAM SYSTEM FOR STRATEGY DEVELOPMENT BUILDING THE OPTIMAL FRAMEWORK PARALLEL PROGRAMS

Bakanov V.M., doctor of technical Sciences, professor

National Research University "Higher School of Economics" (HSE), Russia, 101000, Moscow, Myasnitskaya Str. 20

Physicotechnological Institute of Moscow Technological University, Russia, Moscow, 107996, Stromynka, 20.

@ For correspondence: Bakanov V.M., e-mail: e881e@mail.ru

The work is devoted to solving the problem of creating rational methods for developing a skeleton (a plan, a timetable for execution) of parallel programs for real parallel computing systems. To solve this problem, a software environment has been developed that allows implementing various strategies for building skeleton for performing parallel programs and assessing the quality of these strategies. To simulate and optimize methods, the built-in scripting language of Lua programming is used. The results of applying some of the proposed strategies for constructing rational plans for the execution of parallel programs are given.

Key words: graph representations of the algorithm, analysis of the program information structure, a tiered-parallel form of the information graph, rational parameters for executing parallel programs, a strategy for constructing a rational plan for executing a parallel program.

Для решения современных прикладных задач в промышленности требуются все более высокопроизводительные вычислительные системы; в настоящее время из путей повышения производительности доступен метод параллелизации обработки данных, путь же повышения тактовой частоты ограничивается фундаментальными законами и технической реализацией производства процессоров.

Однако параллелизации (фактически одновременное - параллельное выполнение различных частей одной программы на независимо работающих вычислительных блоках) требует от алгоритмистов и программистов дополнительных усилий по выявлению в алгоритме участков, могущих быть исполненными параллельно; основное требование к таким блокам (*зёрнам* или *гранулам*) параллелизма состоит в независимости (ортогональности) их по данным. Анализ алгоритмов (часто именуемый *исследованием тонкой информационной структуры алгоритма*) по такому критерию затруднен, поэтому говорят о скрытом (не фиксируемом при *поверхностном рассмотрении*) параллелизме; мощным инструментом 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 усложняется необходимостью рационального использования ресурсов конкретной многопроцессорной вычислительной системы (МВС). Эти ресурсы (число и возможности параллельно работающего поля вычислителей) всегда ограничены, поэтому задача планирования использования ресурсов столь важна. В настоящее время именно априорному (на этапе компиляции) планированию при реализации программ на МВС отдается приоритет. Ярким примером является создание программ для параллельных вычислителей архитектуры сверхдлинного командного слова (VLIW, *Very Long Instruction Word*) - аппаратно-программная иделогема микропроцессорной архитектуры с явным параллелизмом команд (EPIC, *Explicitly Parallel Instruction Computing*).

Часто для представления и анализа программ используются (наряду с иными формами) информационные графы (напр., проект AlgoWiki, [2]). На практике для максимального по-

вышения быстродействия оптимизации подвергаются критические участки программы, не имеющие ветвлений; в этом случае информационный граф достаточен для описания алгоритма.

Целью AlgoWiki является "дать исчерпывающее описание алгоритма, которое поможет оценить его потенциал применительно к конкретной параллельной вычислительной платформе". Естественным следующим (перед процедурой собственно кодирования) этапом в цикле разработки параллельного приложения должно являться моделирование выполнения программы на конкретной МВС (с учётом параметров вычислительного поля и размерности данных), при этом формируется рациональный каркас (фактически *план выполнения*) параллельной программы (рис.1, этап выделен двойной рамкой). Именно этот этап является предметом рассмотрения данной работы.

Рис. 1. Фрагмент цикла разработки эффективной параллельной программы.

При представлении программы информационным графом (ИГА, вычислительная модель “операторы - операнды”) вершины графа соответствуют преобразователям информации (операторам), а дуги - информационным связям операторов (данным). В данном случае термин "операторы" в известной степени условен, ибо под "операторами" понимаются отдельные последовательности вычислений (*блоки, подзадачи*) любого (в принципе) размера.

Удобным методом выявления параллелизма является представление ИГА в ярусно-параллельной форме (ЯПФ); при этом операторы, могущиеся выполняться независимо друг от друга (фактически параллельно) располагаются на одном уровне (*ярусе*). Глубина ЯПФ (определяемая числом ярусов) задает общее время выполнения алгоритма, ширина ЯПФ – максимальное число задействованных отдельных (параллельно работающих) вычислителей (напр., отдельных ядер процессора или узлов) данной МВС.

Собственно каноническая ЯПФ может уже рассматриваться как некий *план выполнения* (*каркас*) частей параллельной программы. Недостатком при этом является значительный разброс ширин ярусов ЯПФ. Подобный режим реализуем лишь на гипотетической МВС с бесконечно большим числом параллельных вычислителей (согласно *концепции неограниченного параллелизма*, [1]).

В большинстве ЯПФ имеется *вариативность в расположении вершин графа* (операторов) между ярусами (возможность перемещения операторов между ярусами, которая огра-

ничивается соблюдением информационных зависимостей между операторами); для ЯПФ графа на рис.2 слева оператор 5 может быть перемещен "вниз" на любой ярус с 2 по 5-й, оператор 11 - на любой ярус с 2 по 4-й, оператор 7 - на ярус 2. Эта особенность ЯПФ даёт возможность оптимизации ЯПФ (в смысле, напр., достижения наибольшей равномерности распределения операторов по ярусам); в случае рис.2 слева возможна "разгрузка" яруса 1 от операторов 5 и 11, в результате приведенная программа может быть выполнена на 2 (вместо 4-х) параллельно работающих вычислителях за то же время. В то же время встречаются ЯПФ с нулевой вариативностью (рис.2 справа).

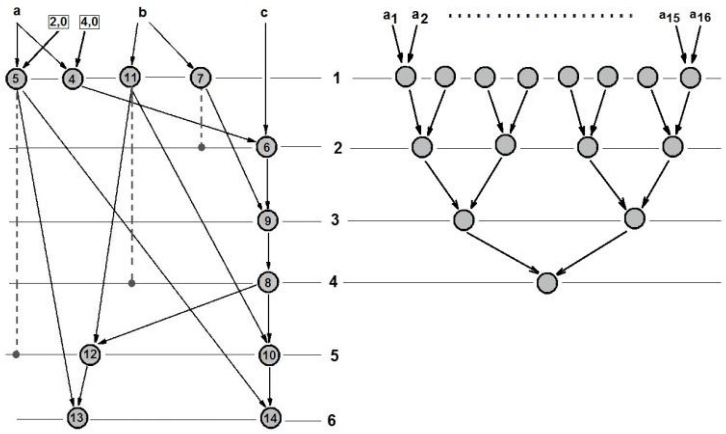


Рис. 2. Примеры ярусно-параллельной формы алгоритмов с потенциалом вариативности расположения операто-ров по ярусам (слева, процедура вычисления корней полного квадратного уравнения, пунктиром показан допустимый диапазон расположения операторов по ярусам) и с нулевым потенциалом вариативности (справа, процедура сдваивания для 16 чисел); цифры в центре - номера ярусов ЯПФ.

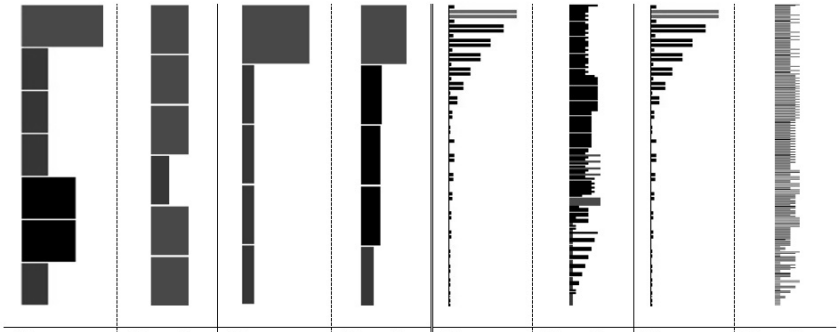
Общая задача близка к проблеме разбиения графов, относится к классу *NP*-полных и решаться может методом разработки эвристических алгоритмов методов.

Для реализации и оптимизации эвристик поиска рациональных (стремящихся к оптимальным) планов выполнения параллельных частей параллельных программ разработано специализированное программное обеспечение инструментального уровня (*программный стенд*), в котором для описание стратегий решения задачи использован встроенный скриптовый язык программирования Lua [3]. Клиентская часть программной системы (предварительное сообщение приведено в работе [4]) написана на языке программирования C++ , является 32-битным GUI-приложением Windows (рис.3) и доступна для свободной выгрузки с ресурса http://vbakanov.ru/spf@home/ (WEB-сайт автора статьи).

Набор API-вызовов разработанной системы позволяет реализовывать любой из возможных критериев оптимизации (а также их комбинации), т.к. выбор критерия осуществляет собственно пользователь (на основе задач, им поставленных).

В случае применение данной системы в качестве компоненты распараллеливающего компилятора граф алгоритма априорно существует (компилятор при работе производит анализ программы на информационные зависимости - т.е. фактически строит граф). Автор для получения корректного ИГА ряд лет использует программный симулятор вычислителя потоковой (DATA-FLOW) архитектуры [5], в котором программа отлаживается (включая количественное решение) и в дальнейшем экспортируется в файловый формат списка смежных вершин.

На рис.3 в графической интерпретации (ленточные диаграммы распределения числа операторов по ярусам) приведены результаты применения различных стратегий преобразования ЯПФ для двух случаев - стратегии #01 (перемещение операторов с наиболее нагруженных на наименее нагруженные яруса при условии сохранения высоты ЯПФ, критерий остановки алгоритма – перебор всех операторов, могущих быть перенесенными “с холмов во впадины” с целью балансировки ЯПФ) и стратегии #02 (балансировка при условии увеличения высоты ЯПФ, критерий останова – “ужатие” (снижение ширины ЯПФ) до величины, не превышающей заданной) для нескольких ИГА распространенных вычислительных процедур различной сложности.



Исходная ЯПФ (H=6; W=4; k=4; σ=1,07; $\overline{W}$=1,83; CV=0,58)	Преобразованная ЯПФ (H=6; W=2; k=2; σ=0,373; $\overline{W}$=1,83; CV=0,2)	Исходная ЯПФ (H=5; W=125; k=5; σ=40; $\overline{W}$ =45; CV=0,89)	Преобразованная ЯПФ (H=5; W=85; k=3,8; σ=25,3; $\overline{W}$=45; CV=0,56)	Исходная ЯПФ (H=63; W=90; k=90; σ=21,8; $\overline{W}$=12,8; CV=1,7)	Преобразованная ЯПФ (H=141; W=10; k=10; σ=2,69; $\overline{W}$=5,71; CV=0,47)	Исходная ЯПФ (H=63; W=90; k=90; σ=21,8; $\overline{W}$=12,8; CV=1,7)	Преобразованная ЯПФ (H=230; W=5; k=5; σ=1,15; $\overline{W}$=3,50; CV=0,33)
а) Вычисление вещественных корней полного квадратного уравнения		б) Умножение квадратных матриц пятого порядка классическим методом		в) Решение системы линейных уравнений (СЛАУ) десятого порядка методом Гаусса (W_0=10)		г) 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. 3. Диаграммы распределения операторов по ярусам для исходных и преобразованных ЯПФ с помощью различных стратегий преобразования для разных алгоритмов (вариант гомогенного поля параллельных вычислителей).

Видно, что примененные стратегии преобразования ЯПФ снижают коэффициент вариации CV конкретных задач в 2-5 раз для случаев рис.3 а-г) соответственно.

Проведенные вычислительные эксперименты показывают, что ранее описанные (но не являющимися излишне изощренными) стратегии в самом деле позволяют снизить неравномерность ширин ЯПФ (а значит, и уровень дисбаланса вычислительной нагрузки данной МВС) при заданных ограничениях, однако с разной эффективность для различных ИГА. В целом наблюдается повышение эффективности стратегии при увеличении сложности (и, соответственно, вариативности) ИГА.

Результаты исследований применимы для анализа алгоритмов (в плане выявления эффективности выполнения параллельных программ на конкретных МВС), 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. [Электронный ресурс]. Дата обновления: 18.10.2016. URL: http://algowiki-project.org (дата обращения: 21.01.2017).
3. Иерузалимски Роберту. Программирование на языке Lua. — М.: ДМК Пресс, 2014. — 382 с.
4. Баканов В.М. Программный инструментарий анализа информационной структуры алгоритмов по их информационным графам. // Параллельные вычислительные технологии (ПаВТ’2016): труды международной научной конференции (г. Архангельск, 28.03–01.04.2016). — Челябинск: Издательский центр ЮУрГУ, 2016, с. 432-441.
5. Баканов В.М. Управление динамикой вычислений в процессорах потоковой архитектуры для различных типов алгоритмов. // Журнал "Программная инженерия", 2015, № 9, с. 20-24.