УДК 004.4'2 + 004.51

В.М.Баканов,

д-р техн. наук, профессор, vbakanov@hse.ru

Национальный Исследовательский Университет

"Высшая Школа Экономики" (НИУ ВШЭ), Россия, Москва

ФОРМИРОВАНИЕ ПЛАНА ВЫПОЛНЕНИЯ ЧАСТЕЙ ПАРАЛЛЕЛЬНОЙ ПРОГРАММЫ И ПРОГРАММНЫЙ СТЕНД ДЛЯ РАЗРАБОТКИ РАЦИОНАЛЬНЫХ СТРАТЕГИЙ ПЛАНИРОВАНИЯ

В обозримом будущем реальным путем повышения производительности вычислительных систем является подход на основе распараллеливания процессов обработки данных (далее - параллелизация). Путь повышения тактовой частоты процессоров ограничивается технологией их производства и фундаментальными законами.

Параллелизация (возможность одновременного, параллельного выполнения различных частей одной программы на независимо работающих вычислителях) требует от алгоритмистов и программистов дополнительных усилий не только по выявлению в алгоритме участков, которые могут быть исполнены параллельно, но и решения более сложной задачи рационального использования оборудования конкретной многопроцессорной вычислительной системы (МВС).

Анализом алгоритмов на наличие потенциала параллелизма, часто именуемый *исследованием тонкой информационной структуры алгоритма*, в нашей стране занимались В.В.Воеводин и Вл.В.Воеводин [1]. Конструктивным подходом при таком анализе является представление программ графовыми моделями.

Несмотря на наличие архитектур параллельных вычислительных систем, не нуждающихся в априорном планировании (напр., потоковый вычислитель

аппаратно определяет очередность выполнения операторов на этапе выполнения программ), именно априорное планирование в настоящее время приоритетно.

Выбор модели графового представления обуславливается компромиссом между требованиями к глубине постановки задачи и сложностью обработки графа. Как правило, для представления и анализа программ используются (наряду с иными формами) информационные графы (напр., проект AlgoWiki, [2]). При таком подходе для максимального повышения быстродействия оптимизации подвергаются 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 платформе" [2]. Следующим (перед процедурой собственно кодирования) и естественным этапом в цикле разработки параллельного приложения должно являться моделирование выполнения программы на конкретной МВС с целью разработки плана выполнения (каркаса) параллельной программы (рис. 1, рассматриваемый в данной работе этап выделен серым фоном).

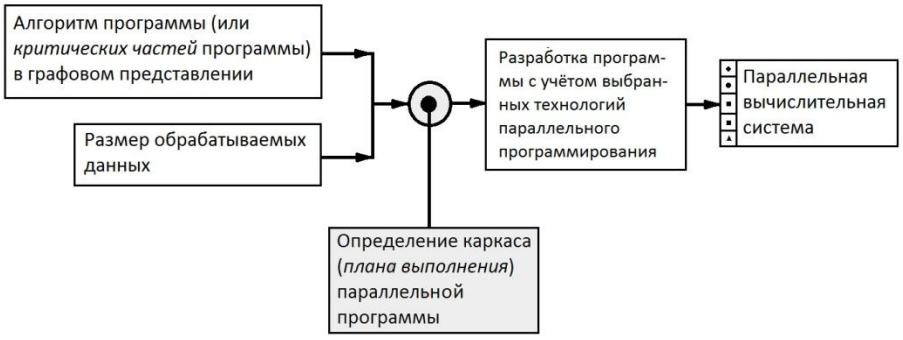

Рисунок 1. Фрагмент цикла разработки эффективной параллельной программы

При представлении программы информационным графом алгоритма (ИГА, вычислительная модель “операторы - операнды”) вершины графа соответствуют преобразователям информации (операторам), а дуги - информационным свя-

зям операторов (операндам). Под "операторами" понимаются отдельные последовательности вычислений (гранулы параллелизма) любого (в принципе) размера (в зависимости от соглашений, принятых при декомпозиции исходной программы). При оптимизации каркаса параллельной программы для процессора со сверхдлинным машинным словом оператором служит собственно машинная инструкция (параллелизм уровня машинных команд), для вычислительных кластеров под оператором следует понимать блок команд, выполняемых на каждом узле кластера.

В общем случае дуги ИГА нагружены величинами объема пересылаемых данных, вершины - значениями вычислительной сложности операторов (в машинных командах или тактах процессора). Время пересылки данных определяется технологией обмена с учетом латентности коммуникационной среды, а время выполнения операторов - параметрами конкретного вычислителя (в общем случае вычислительное поле МВС гетерогенно).

Удобным и полезным методом выявления параллелизма является представление ИГА в ярусно-параллельной форме (ЯПФ). Формально каждый ярус ЯПФ является сечением ИГА, удовлетворяющим условию независимости по данным находящимся на этом ярусе операторов.

Глубина ЯПФ (число ярусов, длина критического пути графа) определяет общее время выполнения алгоритма, ширина ЯПФ – максимальное число задействованных отдельных (параллельно работающих) вычислителей данной МВС. В зависимости от начала построения ЯПФ (от исходных к выходным данным или наоборот) получаем "верхний" или "нижний" канонические варианты ЯПФ, два таких представления ограничивают весь диапазон возможного расположения операторов на ярусах ЯПФ.

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 параллелизма*). Реальные ЯПФ информационных графов обладают большой неравномерностью в распределении числа операторов по ярусам (показательный пример - применяемый для ассоциативных операций *граф сдваивания*). 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).

На рис. 2 слева показана ЯПФ графа простой программы вычисления вещественных корней полного квадратного уравнения и (справа) ЯПФ процедуры сдваивания.

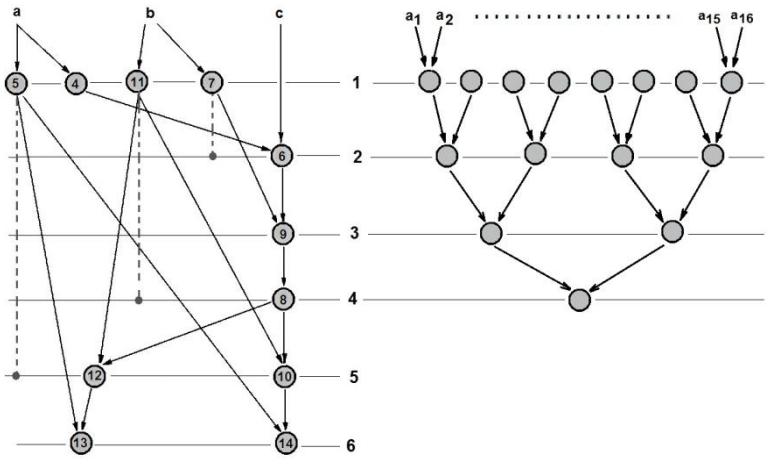

Рисунок 2. Примеры ЯПФ алгоритма с имеющимся потенциалом вариативности расположения операторов по ярусам (слева, пунктиром показан допустимый диапазон расположения операторов по ярусам) и с нулевым потенциалом вариативности (справа); цифры в центре - номера ярусов ЯПФ.

Для количественной оценки уровня вариативности (возможности перемещения операторов с уровня на уровень без нарушения информационных связей) ***V*** конкретной ЯПФ предложено использовать (в первом приближении) выражения:

$$V_P = \sum \frac{P_i}{P} \qquad (1)$$

и

$$V_T = \sum \frac{T_i^{max} - T_i^{min}}{T}, \qquad (2)$$

где в обоих случаях суммирование ведется по номерам операторов i, P_i - число обладающих вариативностью операторов, P - общее число операторов, T_i^{max} и T_i^{min} - максимальный и минимальный номера ярусов данной ЯПФ, на которых может располагаться i-тый оператор (фактически диапазон возможного их расположения), T - общее число ярусов ЯПФ. Выражения (1) учитывает вклад числа операторов со свойством вариабельности, выражение (2) - величину собственно вариабельности, оба они дают ноль при отсутствии вариабельности и монотонно возрастают с её увеличением; разумна также суперпозиция выражений (1,2) в форме их произведения.

Для ЯПФ графа на рис. 2 слева возможна перестановки операторов с яруса на ярус, в результате чего программа может быть выполнена на 2 (вместо 4-х) параллельно работающих вычислителях за то же время. При этом изменятся накладные расходы на хранение и передачу данных, что также может быть учтено.

Данная задача близка к проблематике разбиения графов и, согласно [3], относится к классу *NP*-полных. В этом случае неизбежно применение эвристических методов; с учетом этого корректнее говорить о нахождении не оптимального, а *рационального* решения, приближающегося к оптимальному. Областью оптимизации служит все множество планов выполнения операторов параллельной программы (с учетом сохранения информационных связей). В качестве критерия оптимизации разумно принимать максимум скорости выполнения и/или полноту использования существующего оборудования. 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 улучшение критерия оптимизации) - от "быстрых" методов (основанных, напр., на использовании ЯПФ с последующими преобразованиями) до более трудоемких

(методы "ветвей и границ", эволюционные и др.) для получения лучших решений. Трудоемкие методы более подходят для лабораторных исследований, для использования в составе компиляторов полезными будут именно "быстрые" методы.

Таким образом можно определить направление данного исследования как разработку "быстрых" эвристик решения задачи получения рациональных (стремящихся к оптимальным) планов выполнения частей параллельной программы.

Будем использовать следующие обозначения параметров ЯПФ информационного графа:

- H - глубина ЯПФ, определяется количеством ярусов в нем
- W_i – ширина i-того яруса (количество операторов на нём)
- W – ширина ЯПФ графа, определяется максимумом операторов по всем ярусам, $W = max(W_i)$
- $\overline{W}$ – средняя ширина ЯПФ, $\overline{W} = \frac{1}{N}\sum_{i=1}^{i=N} W_i$, N - число ярусов ЯПФ
- k - степень (коэффициент) неравномерности распределения операторов по ярусам ЯПФ графа, $k = max(W_i)/min(W_i)$
- σ - среднеквадратичное отклонение ширин ярусов, σ=$\sqrt{\frac{1}{N}\sum_{i=1}^{i=N}(W_i - \overline{W})^2}$
- CV - коэффициент вариации ширин ярусов ЯПФ, $CV = {}^{\sigma}/_{\overline{W}}$.

Исходя из общего понимания процесса выполнения параллельной программы на заданном вычислительном поле можно сформулировать перечисленные далее два типовых направления оптимизационной задачи:

а) Требование максимальной производительности (условие неувеличения числа ярусов канонической ЯПФ), при этом использование имеющихся параллельных вычислителей может быть неэффективным (в идеальном случае число вычислителей не менее $\overline{W}$, в реальности зависит от свойств алгоритма и качества алгоритма "балансировки" операторов по ярусам). Следует заме-

тить, что потенциал оптимизационной задачи в этой постановке сильно ограничен для большинства алгоритмов.

б) Наиболее полное использование доступных ресурсов (количества выделенных данной задаче вычислительных узлов) имеющейся МВС при допущении увеличения времени выполнения. Для наглядности полезно представить этот процесс как подготовку связки (*bundle*), состоящей из не более чем заданного числа 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*).

Заметим, что в данном случае оптимизируется выполнение конкретной программы, в случае многозадачного режима необходима оптимизация по всем одновременно выполняемым заданиям.

Известен ряд программных продуктов, близких по постановочным целям, функциональные возможности которых для данного случая явно избыточны.

Для реализации и оптимизации эвристических алгоритмов поиска рациональных (стремящихся к оптимальным) планов выполнения частей параллельных программ разработано специализированное программное обеспечение инструментального уровня, в котором для описание стратегий решения задачи использован встроенный скриптовый язык программирования Lua по причине его известных полезных качеств [4].

Клиентская часть программного инструментария (предварительное информационное сообщение приведено в работе [5]) разработана на языке программирования C++, является 32-битным GUI-приложением Windows и доступна для свободной выгрузки с WEB-ресурса автора http://vbakanov.ru/spf@home/content/spf@home.rar.

Набор API-вызовов разработанной системы позволяет реализовывать любой из возможных критериев оптимизации (включая их комбинации), т.к. вы-

бор критерия осуществляет собственно пользователь (на основе задач, им поставленных). Например, совсем необязательно использовать представление ИГА в виде ЯПФ (хотя API системы включает такой вызов).

Требование 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 модели системой тегов (дополнительных параметров) для операторов и вычислителей. Также реализована система тегов для описания мер вершин и дуг ИГА (фактически операторов и линий передачи сообщений), позволяющая задавать для них собственные метрики.

Для случая гетерогенности поля параллельных вычислителей предложена (и реализована на Lua) стратегия, позволяющая разбивать ярусы ЯПФ на подъярусы. Если в пределах данного яруса имеется операторов не более чем число вычислителей, могущих выполнять операторы этого типа, то они выполняются параллельно, в ином случае эти операторы вынужденно выполняются последовательно на подъярусах (рис.3).

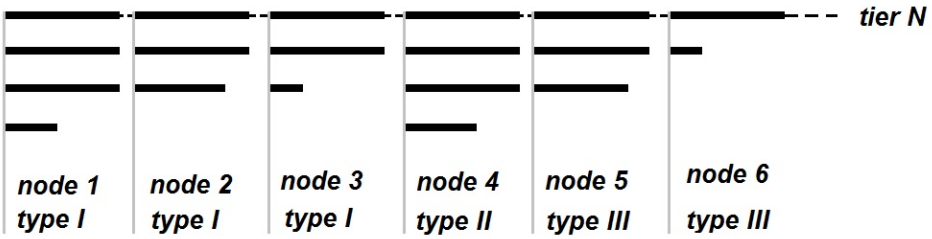

Рис. 3. Выполнение параллельной программы на гетерогенном поле вычислителей согласно схемы расщепления ярусов для 6 параллельных вычислителей (*node*) трех типов - *type I, type II* и *type III* по 3, 1 и 2 штуки соответственно для *N*-ного яруса (*tier*) ЯПФ.

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

$$T = \sum_{j} \left(\max_{k_j} \sum_{i} t_{ik} \right) \qquad (3)$$

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

Задача минимизации общего времени решения усложняется в случае возможности выполнения каждого оператора на нескольких вычислителях вследствие неоднозначности t_{ik} в выражении (3); здесь необходима дополнительная балансировка выполнения по подъярусам.

Рассмотрим две (не излишне изощренные) стратегии преобразования ЯПФ - стратегия с метафорическим названием БУЛЬДОЗЕР (перемещение операторов с наиболее нагруженных на наименее нагруженные яруса при условии сохранения высоты ЯПФ, критерий остановки алгоритма – перебор всех операторов, которые могут быть перенесены “с холмов во впадины”) и стратегия ДИХОТОМИЯ (балансировка при условии увеличения высоты ЯПФ, критерий останова - снижение ширины ЯПФ до величины, не превышающей заданной).

Эффективность этих стратегий проверялась на распространенных алгоритмах численной обработки (вследствие чрезмерного объема данных приводятся лишь общие результаты). При применении стратегии БУЛЬДОЗЕР снижение показателей k и CV достигает 1,5-2 раз (большие значения соответствуют обработке данных больших размеров), применение стратегии ДИХОТОМИЯ характеризуется значениями тех же показателей до 10 (в некоторых случаях много больше - в зависимости от заданных условий и конкретных алгоритмов).

Особенна интересна и практически важна задача априорного (до проведения собственно преобразований ЯПФ) предсказания уровня эффективности стратегий (задача распознавания ситуации и применение методов искусственного интеллекта).

В ходе исследований в целом удалось добиться поставленной цели - разработки рациональных стратегий составления плана выполнения частей параллельных программ, использование предложенного программного стенда оказалось эффективным. Получены количественные параметры текущего приближения при совершенствовании стратегий оптимизации выполнения параллельных программ, выявлены пути дальнейшего развития таких стратегий.

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. [Электронный ресурс]. Дата обновления: 18.10.2016. URL: http://algowiki-project.org (дата обращения: 01.05.2017).
3. Гери М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982. — 416 с.
4. Иерузалимски Роберту. 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): труды международной научной конференции (г. Архангельск, 28.03–01.04.2016). —Челябинск: Издательский центр ЮУрГУ, 2016, с. 432-441.

Аннотация: Анализируются 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 (стратегий). 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.

Abstract: Approaches to the development of rational (aspiring to optimal) methods for constructing plans for executing applied computing tasks on real parallel computing systems (frameworks for execution), as well as an instrumental software environment for implementing such methods (strategies) are analyzed. The criteria and parameters for optimizing planning methods are proposed. The results of applying some developed strategies for constructing rational plans for parallel programs are presented.

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