УДК 519.874

ЭВРИСТИЧЕСКИЙ МЕТОД ДЛЯ РЕШЕНИЯ ЗАДАЧИ РАСПАРАЛЛЕЛИВАНИЯ АЦИКЛИЧЕСКОГО АЛГОРИТМА НА МНОГОПРОЦЕССОРНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ

В.В. Слободской

Нижегородский госуниверситет им. Н.И. Лобачевского

slvital@gmail.ru

Поступила в редакцию 21.05.2010

Рассматривается задача распараллеливания алгоритма, реализуемого посредством канонической совокупности взаимозависимых операций, моделируемой ориентированным взвешенным ациклическим графом. Для решения задачи предлагается эвристический алгоритм, основанный на вычислительных процедурах метода ветвей и границ.

Ключевые слова: метод ветвей и границ, ациклический алгоритм, задача распараллеливания, каноническая совокупность взаимозависимых операций, многопроцессорная вычислительная система.

Введение

Использование точных методов для решения задачи распараллеливания алгоритма становится невозможным, если алгоритм состоит хотя бы из сотни операций. В то же время простые эвристические алгоритмы зачастую позволяют найти лишь некоторое решение, далёкое от оптимального. Мы попытаемся разработать алгоритм, дающий лучшие результаты за приемлемое время, чем простые жадные алгоритмы.

Как и в [1], будем рассматривать алгоритмы, представимые в виде канонической сети взаимозависимых операций. Такими алгоритмами являются алгоритмы без циклов и ветвлений. Вычислительная система – это набор вычислительных элементов (процессоров), обладающих различными скоростями выполнения операций. Предполагается, что известной является матрица скоростей передачи данных от одного процессора другому.

В отличие от [1, 2] в этой работе для решения задачи распараллеливания рассматривается эвристический метод, использующий идеологию метода ветвей и границ, дающий существенно лучшие результаты по сравнению с фронтальным алгоритмом. Также приводятся некоторые модификации метода, позволяющие уменьшить множество поиска при поиске оптимального решения. В отличие от [3] в этой работе рассматривается общая модель задачи, позволяющая моделировать выполнение операций на вычислительных системах.

Задача распараллеливания ставится как оптимизационная задача на некотором множестве допустимых решений. Область исследований распараллеливания задач на вычислительных системах, состоящих из нескольких вычислительных элементов с разными скоростями, с помощью решения оптимизационных задач разбивается на две части – распараллеливание независимых операций и распараллеливание операций, зависимости между которыми представимы в виде канонической сети. Случай независимых операций достаточно хорошо исследован. В [4, 5] предлагаются приближённые алгоритмы решения подобной задачи с приводимыми оценками погрешности. Работа [6] предлагает 2-приближённый алгоритм ее решения, также приводится и доказывается теорема о том, что для подобной задачи не существует полиномиального ε-приближенного алгоритма с $\varepsilon < 3/2$. В [7] приводятся приближенные алгоритмы для задач как с независимыми, так и с зависимыми операциями. В работе [8] приводится аналог фронтального алгоритма для задачи с зависимыми операциями. В [9] рассматривается оптимальный алгоритм для решения такой задачи при условии малого количества операций.

Цель данного исследования – создание средства, способного моделировать выполнение рассматриваемого типа алгоритмов на любой вычислительной системе. Цель данной работы – создание эффективного алгоритма, решающего задачу распараллеливания.

Содержательное описание задачи распараллеливания

Имеется вычислительная система, состоящая из n вычислительных элементов (процессоров),

объединённых в сеть таким образом, что имеется возможность передачи данных от каждого вычислительного элемента любому другому. Скорость передачи данных между процессорами задаётся в виде матрицы. Каждый процессор имеет скорость вычисления операции. Считается, что время выполнения любой операции на процессоре обратно пропорционально скорости вычисления процессора. Скорость самого медленного процессора принимается равной единице.

Задан алгоритм, представляющий собой каноническую сеть взаимозависимых операций, где каждая операция (кроме начальных) имеет набор предшествующих операций и каждая операция (кроме завершающих) имеет набор последующих операций. Операция не может начинать выполняться, пока не завершены все предшествующие ей операции. Для каждой операции известно ее время выполнения на самом медленном процессоре. Для любой операции и любой последующей для неё операции задана величина, определяющая количество данных, которые необходимо передать для выполнения последующей операции. Операция на процессоре не может начинаться, пока на этот процессор не переданы данные со всех операций, предшествующих этой операции. Передача данных может начинаться только после завершения операции. Считается, что данные на процессор можно передавать независимо от его работы, т.е. параллельно с текущими вычислениями, не замедляя при этом его работу. Также считается, что выполнение более одной операции на процессоре является невозможным, поэтому каждый процессор в одно и то же время может обрабатывать лишь одну операцию.

Необходимо таким образом назначить операции на процессоры, чтобы общее время завершения обработки всех операций было минимальным.

Общая математическая модель

Исходные параметры математической модели

Пусть J – множество всех операций, которые необходимо выполнить, I – множество процессоров вычислительной системы, $K(j)$ – множество операций, непосредственно предшествующих операции с номером j, $K(j) \subset J$, $j \in J$; t_j – время выполнения операции j на процессоре со скоростью 1, $t_j > 0, j \in J$; p_i – скорость процессора i, $p_i > 0$, $i \in I$ (под скоростью процессора здесь будем понимать величину, определяющую во сколько раз действительная скорость процессора больше действительной скорости самого медленного процессора).

Пусть $\|M\|$ – матрица размерности $|J|\times|J|$, элемент которой m_{kl} определяет количество данных, которые должны быть переданы после выполнения операции k для выполнения операции l, $k \in K(l)$, $k,l \in J$, $m_{kl} = 0$, если $k \notin K(l)$; $\|S\|$ – матрица размерности $|I|\times|I|$, элемент которой s_{uv} определяет скорость передачи данных от процессора u к процессору v, $s_{uv}>0$, $u,v \in I$ (здесь предполагается, что скорость передачи данных с одного процессора на самого себя определяется очень большим числом); $Q = \{(j,k)\}$, где j – непредшествующая операция для k, k – непредшествующая операция для j; J^0 – множество начальных операций (не имеющих предшествующих) $J^0 \subseteq J$; J^D – множество завершающих операций, $J^D \subseteq J$.

Варьируемые параметры математической модели

В качестве неизвестных математической модели определим $|J|$-мерный вектор $\vec{x}$, компонента которого x_j определяет время начала выполнения операции j, $j \in J$, и булеву матрицу $\|Y\|$ размерности $|J|\times|J|$, элемент которой $y_{ji} = 1$, если операция j выполняется на i-м процессоре, $j \in J$, $i \in I$; $y_{ji} = 0$ – в противном случае.

Ограничения математической модели

технологическое ограничение:

$$x_k \geq \max_{j \in K(k)} \left(x_j + t_j \sum_{l \in I} \frac{y_{jl}}{p_l} + m_{jk} \sum_{i \in I} \sum_{l \in I} \frac{y_{ji} y_{kl}}{s_{il}}\right), k \in J; \quad (1)$$

невозможность выполнения двух операций на одном процессоре одновременно:

$$x_k \geq x_j + t_j \sum_{l \in I} \frac{y_{jl}}{p_l} \quad \text{или}$$

$$x_j \geq x_k + t_k \sum_{l \in I} \frac{y_{kl}}{p_l}, \text{ если } (j,k) \in Q, y_{jl} = y_{kl}, \quad (2)$$

для всех $l \in I$; $k, j \in J$;

каждая операция должна выполняться на одном процессоре:

$$\sum_{l \in I} y_{jl} = 1, \; j \in J; \quad (3)$$

ограничения для начальных операций:

$$x_j \geq 0, \; j \in J^0 \quad (4)$$

Таблица 1

Вычислительный эксперимент

Алгоритм / Задача	Фронт/А	Фронт/Б	ЭМ
10 КР	11 ч 00 мин 00 с / 00:00	7 ч 17 мин 8 с / 00:00	7 ч 08 мин 34 с / 00:00
10 БКР	14 ч 13 мин 12 с / 00:00	16 ч 10 мин 54 с / 00:00	7 ч 12 мин 0 с / 00:00
15 КР	36 ч 0 мин 0 с / 00:00	36 ч 0 мин 0 с / 00:00	18 ч 37 мин 30 с / 00:00
15 БКР	34 ч 59 мин 40 с / 00:00	34 ч 59 мин 40 с / 00:00	10 ч 24 мин 0 с / 00:00
20 КР	19 ч 15 мин 0 с / 00:00	18 ч 15 мин 0 с / 00:00	15 ч 36 мин 0 с / 00:00
20 БКР	66 ч 26 мин 47 с / 00:00	66 ч 26 мин 47 с / 00:00	17 ч 10 мин 02 с / 00:00
100 КР	40 ч 21 мин 27 с / 00:00	37 ч 6 мин 37 с / 00:00	35 ч 38 мин 06 с / 07:44
100 БКР	503 ч 23 мин 56 с / 00:00	186 ч 33 мин 48 с / 00:00	76 ч 50 мин 27 с / 01:17
300 КР	242 ч 50 мин 34 с / 00:00	185 ч 34 мин 15 с / 00:00	111 ч 41 мин 54 с / -------*
300 БКР	1354 ч 37 мин 24 с / 00:00	722 ч 36 мин 38 с / 00:00	387 ч 50 мин 32 с / -------*
1000 КР	2260 ч 26 мин 12 с / 00:00	2085 ч 3 мин 59 с / 00:00	1151 ч 57 мин 52 с / -------*
1000 БКР	2196 ч 49 мин 50 с / 00:00	1544 ч 25 мин 51 с / 00:00	1172 ч 08 мин 06 с / -------*
10000 КР	2903 ч 20 мин 58 с / 00:09	2447 ч 33 мин 38 с / 00:22	2333 ч 00 мин 00 с / -------*

естественные ограничения на варьируемые переменные:

$$y_{ji} \in \{0,1\}, j \in J, i \in I, \quad x_j \geq 0, j \in J. \qquad (5)$$

Задача минимизации общего времени выполнения операций

Требуется найти такое решение системы ограничений (1)–(5), на котором принимает минимальное значение критерий, определяющий общее время выполнения операций:

$$F\ (\vec{x}, Y) = \max_{j \in J^D}(x_j + t_j \sum_{l \in I} \frac{y_{jl}}{p_l}). \qquad (6)$$

Эвристический метод для решения задачи распараллеливания

Дискретность переменных и существенная нелинейность ограничений (2) в случае большого числа переменных и ограничений, соответствующих реальным вычислительным задачам, не позволяют для решения поставленной задачи использовать точные методы. Рассмотрим эвристику, использующую идеологию метода ветвей и границ, позволяющую находить близкие к оптимальному решения за незначительное время (в дальнейшем – эвристический метод).

Процедура ветвления для рассматриваемой задачи заключается в построении дерева ветвления по следующей схеме. В качестве корневого узла дерева ветвления рассматривается решение, в основе которого нет ни одной включенной операции. К этому узлу добавляются дочерние узлы в количестве всех возможных комбинаций назначения начальных операций на каждый процессор. Далее к каждому узлу дерева ветвления добавляются дочерние узлы в количестве всех возможных комбинаций назначения всех последующих операций для операций, уже назначенных для этого узла, на каждый процессор. Процесс повторяется до тех пор, пока не будут достигнуты листовые узлы дерева, в которых уже назначены все операции задачи.

При таком ветвлении вполне вероятен случай, когда два разных узла дерева ветвления представляют собой два одинаковых частичных расписания (отличаются лишь порядком назначения операций), вследствие чего один из таких узлов может быть удалён из дерева без потери оптимального решения.

Однако на задачах большой размерности вычисление всех узлов не представляется возможным. Для ускорения поиска хорошего решения на каждом шаге ветвления ограничим очередь обработки узлов некоторым значением H. Для этого после обработки всех узлов предыдущего шага удалим из очереди некоторые узлы, чтобы общее их количество не превышало H. Такая модификация операции делает алгоритм эвристическим, но при этом оставляет шанс найти оптимальное решение и даёт ряд преимуществ:

– время работы алгоритма ограничено величиной $H \mid J \mid t_{уз}$, где $t_{уз}$ – время обработки первого узла дерева ветвления;

– потребление памяти ограничено вследствие ограничения очереди;

– есть возможность использовать различные схемы удаления узлов из очереди.

В качестве одной из таких схем предлагается рассмотреть схему сохранения наиболее перспективных вариантов: в очереди сохраняются $H/2$ узлов с лучшей нижней оценкой и $H/2$ узлов с лучшей верхней оценкой из оставшихся. Однако такая схема не имеет шансов найти оптимальное решение в случаях, когда оно изначально находится в ветке неперспективного направления. Для исключения такого варианта предлагается ввести некоторое число P и сохра-

Таблица 2

Процент отклонения «рекорда» относительно максимальной нижней оценки

Задача	МНО	Фронт/А	Фронт/Б	ЭМ
10 КР	7 ч 08 мин 34 с	54	1.99	0
10 БКР	7 ч 12 мин 0 с	97.5	124.74	0
15 КР	18 ч 37 мин 30 с	93.2	93.2	0
15 БКР	10 ч 24 мин 0 с	70.28	70.28	0
20 КР	15 ч 4 мин 36 с	27.68	21.04	3.47
20БКР	16 ч 29 мин 24 с	302.9	302.9	4.1
100 КР	35 ч 6 мин 0 с	11.79	2.72	1.50
100 БКР	55 ч 54 мин 0 с	800.53	233.74	37.46
300 КР	110 ч 13 мин 20 с	120.32	68.36	1.34
300 БКР	305 ч 48 мин 0 с	342.97	136.3	26.82
1000 КР	1146 ч 13 мин 20 с	97.2	81.9	0.50
1000 БКР	484 ч 30 мин 00 с	353.42	218.76	141.93
10000 КР	2313 ч 1 мин 20 с	25.52	5.81	0.86

нять в очереди $(H - P) / 2$ узлов с лучшей нижней оценкой, $(H - P) / 2$ узлов с лучшей верхней оценкой из оставшихся, и P узлов выбирать из оставшихся случайным образом.

Для формализации процедуры оценок введем дополнительные обозначения. Пусть U – множество вершин дерева ветвления, $S(u)$ – множество операций, ещё не включенных в решение для вершины u, $u \in U$, $M(u)$ – множество операций, для которых включены в решение все предшествующие операции $M(u) \subseteq S(u)$, $u \in U$.

В качестве верхней (достижимой) оценки значения критерия оптимальности принимается значение критерия на решении задачи фронтальным алгоритмом [1]. Для определения нижней оценки рассмотрим величину:

$$H = \frac{\sum_{j \in J} t_j}{\sum_{i \in I} p_i}. \qquad (7)$$

В [1] показано, что для любого решения задачи (1)–(6) $(\vec{x}', Y')$ выполняется условие

$$F(\vec{x}', Y') \geq \frac{\sum_{j \in J} t_j}{\sum_{i \in I} p_i}.$$

В процессе вычислений оценку (7) можно улучшать, учитывая те простои процессоров, которые уже невозможно занять выполнением операций. Тогда

$$H_2 = \frac{\sum_{j \in J} t_j + t^n(u)}{\sum_{i \in I} p_i}, u \in U, \qquad (8)$$

где $t^n(u)$ – суммарное количество простоев процессоров на интервале от начала планирования до минимального времени начала выполнения операций из множества операций, для которых все предшествующие операции уже назначены на процессоры в частичном решении, заданном узлом u.

Экспериментальная часть

Для анализа результатов описанного эвристического метода будем также рассматривать две модификации фронтального алгоритма: Фронт/А – фронтальный алгоритм без схемы сравнения, Фронт/Б – фронтальный алгоритм со схемой упорядочивания операций во фронте по убыванию времени выполнения последующих операций. ЭМ – эвристический метод с параметрами $H = 5$, $P = 1$.

Рассматриваемые задачи можно классифицировать следующим образом:

1. КР – задача с большой конкуренцией за ресурсы, в которой в каждый момент времени во фронте находится большое количество операций, которые могут выполняться.

2. БКР – задача без конкуренции за ресурсы, в каждый момент времени во фронте находится одна-две операции.

Все задачи генерировались случайным образом с помощью генератора задач. Число в указании задачи определяет общее количество операций в задаче (см. табл. 1).

Каждая ячейка содержит два числа – первое – результат работы алгоритма (значение критерия на найденном решении – общее время выполнения всех операций), второе – время работы алгоритма. Метка «-------*» означает, что алгоритм был остановлен по истечении 10 минут, значение в таблице означает полученное к данному моменту решение.

Таблица 2 содержит максимальное среди минимальных значений нижней оценки, полученной различными модификациями МВГ, опи-

санными в [1], в течение 10 минут. Процент отклонения (ПО) вычисляется по следующей формуле: $\text{ПО} = \frac{\text{рекорд} - \text{нижняя оценка}}{\text{нижняя оценка}} 100$.

Результаты

По результатам вычислительного эксперимента видно, что предложенный эвристический метод способен находить существенно лучшие решения, чем фронтальные алгоритмы, за приемлемое время. Относительно низкий процент отклонения от максимальной нижней оценки для данного метода говорит о близости найденных решений к оптимальным.

Заключение

В работе применен новый способ распараллеливания алгоритмов, основанный на решении оптимизационной задачи, поставленной в рамках общей математической модели. Для решения задачи предложен новый эвристический алгоритм, основанный на методе ветвей и границ. Проведён вычислительный эксперимент. Данный подход можно использовать для оценки уровня параллелизма задачи. Варьируя вычислительными ресурсами, можно исследовать поведение алгоритма на различных вычислительных системах.

Список литературы

1. Слободской В.В. Задача распараллеливания ациклического алгоритма // Вестник ННГУ. 2008. №5. С. 113–119.

2. Слободской В.В. Использование метода ветвей и границ для решения задачи распараллеливания ациклического алгоритма // Тезисы докладов конференции «Технологии Microsoft в теории и практике программирования». Н. Новгород: Издательство ННГУ, 2009.

3. Прилуцкий М.Х., Слободской В.В. Распределение ресурсов в задачах балансирования с постоянными длительностями // Вестник ВГАВТ. Межвузовская серия Моделирование и оптимизация сложных систем. 2006. Вып. 1.

4. Jansen K., Porkolab L. Improved Approximation Schemes for Scheduling Unrelated Parallel Machines // Mathematics of Operations Research. 2001. V. 26. № 2. P. 324–338.

5. Serna M., Xhafa F. Approximating Scheduling Unrelated Parallel Machines in Parallel // Computational Optimization and Applications. 2002. V. 21. P. 325–338.

6. Lenstra J.K., Shmoys D.B., Tardos E. Approximation Algorithms for Scheduling Unrelated Parallel Machines // Mathematical Programming. 1990. V. 46. P. 259–271.

7. Samadzadeh F.A., Hedrick G.E. Near-Optimal Multiprocessor Scheduling // Proceedings of the 1992 ACM Annual Conference on Communications. 1992. P. 477–484.

8. Samadzadeh F.A., Hedrick G.E. A Heuristic Multiprocessor Scheduling Algorithm for Creating Near-Optimal Schedules Using Task System Graphs // Proceedings of the 1992 ACM/SIGAPP Symphosium on Applied Computing: Technological Challenges of the 1990`s. 1992. P. 711–718.

9. Sinnen O., Kozlov A.V., Shahul A.Z.S. Optimal scheduling of task graphs on parallel systems // 25th LASTED International Multi-Conference Parallel and Distributed Computing and Networks. 2007.

HEURISTIC METHOD FOR SOLVING ACYCLIC ALGORITHM PARALLELIZATION PROBLEM ON A MULTIPROCESSOR SYSTEM

V.V. Slobodskoy

An algorithm parallelization problem is considered, the algorithm being a canonical set of interdependent operations simulated by a weighted directed acyclic graph. The problem is proposed to be solved by using a heuristic algorithm based on computational procedures of the branch and bound method.

Keywords: branch and bound method, acyclic algorithm, parallelization problem, canonical set of interdependent operations, multiprocessor system.