УДК 681.3.06

И.А. Трещёв

Использование метода перебора последовательностей, как раскраски вершин графа при обходе в ширину, на системах с SMP-архитектурой для решения переборных задач

Предлагаются методы распараллеливания алгоритмов для задач допускающих решение методом перебора с возвратом, ориентированные на использование в системах с SMP-архитектурой. Данные методы применяются для построения многопоточных приложений. Приводятся результаты экспериментального анализа предложенных методов для классических задач.

Ключевые слова: криптография, переборные задачи, рекурсия, многопоточность.

Введение. Криптостойкость популярного алгоритма Диффи-Хеллмана основана на вычислительной сложности обращения функции в некоторой конечной мультипликативной абелевой группе [1]. Перебор возможных решений в данной группе можно осуществлять с помощью метода ветвей и границ. Опишем данный метод и пути его распараллеливания для систем с SMP-архитектурой. Пусть заданы конечные множества $A_1, A_2, \cdots$. Для произвольного целого $n>0$ под n-местным предикатом будем понимать функцию $P: A_1 \times A_2 \times \ldots \times A_n \to \{0,1\}$, принимающую значения 1 (истина) или 0 (ложь). *Предикатом* будем называть произвольную функцию $Q: \bigcup_{n\geq 1} A_1 \times A_2 \times \cdots \times A_n \to \{0,1\}$.

Формулировка задачи. Даны конечные множества $A_1, A_2, \cdots$ и для каждого целого i задан i-местный предикат $P_i(x_1, x_2, \ldots, x_i)$. Предполагается, что для всех $i>1$ справедливы импликации $P_i(x_1, x_2, \ldots, x_i)=1 \Rightarrow P_{i-1}(x_1, x_2, \ldots, x_{i-1})=1$. Требуется построить последовательности $(x_1, x_2, \ldots, x_n)$, где $x_i \in A_i$ при $1 \leq i \leq n$, для которых верны соотношения:

$$P_n(x_1, x_2, \ldots, x_n)=1; \quad (1)$$

$$Q(x_1, x_2, \ldots, x_n)=1. \quad (2)$$

Сведение задачи к раскраске вершин дерева. Обозначим через $R_i \subseteq A_1 \times A_2 \times \ldots \times A_i$ подмножество последовательностей $(x_1, x_2, \ldots, x_i)$, удовлетворяющих условию $P_i(x_1, x_2, \ldots, x_i)=1$. Рассмотрим дерево, вершинами которого являются конечные последовательности $(x_1, x_2, \ldots, x_i) \in R_i$. Корнем служит пустая последовательность. Сыновьями узла $(x_1, x_2, \ldots, x_{k-1})$ будут вершины $(x_1, x_2, \ldots, x_{k-1}, x_k) \in R_k$. Это дерево является подграфом дерева, у которого на уровне $k = 0$ находится корень, и из каждой вершины уровня $k - 1$ выходят $|A_k|$ ребер, имеющих метки $a \in A_k$. Таким образом, посещаются вершины дерева, представленного на рис. 1, что гарантированно приведет к решению.

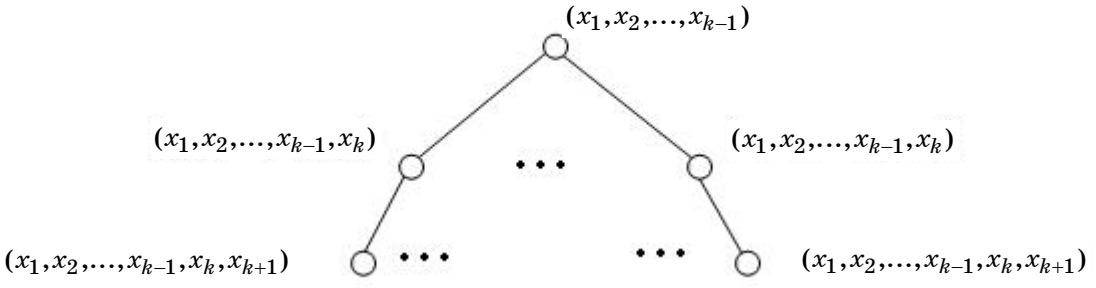

Рис. 1. Поддерево дерева решений

Распараллеливание рекурсивных подпрограмм и параллельная реализация с помощью обхода в ширину. В современных операционных системах возможность потока порождать другие потоки позволяет строить многопоточные программы из рекурсивно

вызываемых подпрограмм. Вместо рекурсивного вызова функций для построения таких программ подставляется создание потоков, соответствующих этим функциям. Более глубокий анализ показывает, что выигрыш в быстродействии при использовании такой схемы отсутствует. Это связано с тем, что при запуске потока мы ожидаем его завершения, не выполняя никаких действий. Отсюда вытекает, что параллельная реализация метода обхода в глубину не дает выигрыша во времени. Тем не менее, перебор последовательностей можно ускорить для систем с SMP-архитектурой. Например, использовать методы, предложенные в работах Н.Е. Тимошевской и В.А. Беляева [3], – методы назначаемых и выделяемых поддеревьев. Для кластерных архитектур при помощи данных методов действительно можно получить ускорение, близкое к теоретически достижимому. Можно также обходить дерево решений в ширину. Заметим, что на k-м ярусе дерева решений может быть запущено не более $|A_k|$ потоков, где $|A_k|$ есть мощность соответствующего множества, причем это возможно лишь при наличии достаточного числа вычислительных узлов. Использование предложенной выше схемы реализации перебора с возвратом как поиска в ширину не совсем оправдано, поскольку «накладные расходы» по времени на создание большого количества потоков могут превзойти выигрыш от использования нескольких процессоров в том случае, если их количество ограничено. Поэтому более рациональное решение заключается в том, чтобы подпрограмма перебора сама определяла, следует ли ей запуститься как потоку, если количество незанятых процессоров отлично от нуля, или как последовательной рекурсивной подпрограммой. Для доступа всех потоков к общей переменной, в которой будет храниться текущее количество занятых процессоров, следует использовать объекты синхронизации [2].

Результаты тестирования многопоточных приложений [4] для классических задач представлены в таблице (время указано в миллисекундах). Тестирование разработанных многопоточных приложений производилось на системе под управлением Microsoft Windows Server 2003, оснащенной двумя четырехъядерным микропроцессорами Intel Pentium 4 Xeon 5320, объем оперативной памяти 8 Gb.

Результаты тестирования многопоточных приложений

Количество потоков	Перебор гамильтоновых циклов в графе	Задача Гаусса о ферзях	Перебор разложений числа в сумму	Перебор возрастающих последовательностей	Генерация разбиений заданного множества
Перебор с возвратом, последовательная реализация					
	13469	23906	5453	18324	18390
Многопоточные приложения, реализующие поиск в ширину как раскраску вершин графа					
1	16172	26953	6297	21125	21344
2	14953	25657	6266	16297	18969
3	13781	24062	6079	15484	18793
4	12640	22219	5625	15188	15188
5	11594	20829	5406	14813	13734
6	10328	18922	5296	13953	12625
7	9453	17047	5235	12578	12422
8	8265	15422	5234	13922	12266
9	7844	14266	5219	14485	12032
10	8297	13078	5187	14875	13500
11	8500	12375	4891	14937	13609
12	12041	12986	4797	15378	17437
13	12437	15187	4735	16225	19422
14	12923	15829	4703	16984	20204
15	13964	16922	4266	17937	22672

Для некоторых задач, как видно из таблицы, значение ускорения пропорционально числу используемых процессоров, но может и не превысить единицы. Пусть искомое решение представлено самым левым путем в дереве решений, тогда параллельный алгоритм, если не учитывать «накладных расходов», осуществит ровно столько же действий, что и последовательный. Противоположная ситуация возникает, если искомое решение представлено самым правым путем.

Литература

1. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. – М.: МЦНМО, 2003. – 328 с.

2. Хусаинов А.А. Архитектура вычислительных систем: Учеб. пособие / А.А. Хусаинов, Н.Н. Михайлова // Комсомольск-на-Амуре: ГОУВПО «КнАГТУ», 2004. – 123 с.

3. Беляев В.А. Распараллеливание обхода дерева поиска для решения задачи о рюкзаке на кластерной системе / В.А. Беляев, Н.Е. Тимошевская // Высокопроизводительные параллельные вычисления на кластерных системах: Матер. междунар. науч.-практ. семинара / под ред. проф. Р.Г. Стронгина. – Нижний Новгород: Изд-во Нижегор. гос. ун-та, 2001. – С. 16–20.

4. Трещев И.А. Построение многопоточных приложений для распараллеливания алгоритмов перебора // Информатика и системы управления. – 2008. – №1 (15). – С. 151–159.

Трещев Иван Андреевич
ГОУВПО «Комсомольский-на-Амуре государственный технический университет», ст. преподаватель кафедры Математическое обеспечение и применение ЭВМ
Эл. адрес: kalkt@yandex.ru

I.A. Treshchov

Using of parallel backtracking, as graph marking in wide tree traverse, on systems with SMP-architecture for solving search problems

Methods of constructing parallel algorithms for tasks, which could be solved with backtrack algorithm, oriented on the SMP-architecture, are proposed in this paper. These methods are used for building multithreaded applications. Experimental results concerning the methods for classic tasks on backtrack are given.

Key words: cryptography, reboric problems, recourse, multithreading.
