O. S. Chernaya, Yu. Yu. Fedorova, S. A. Belyaev

APPLYING OF METHODS AND TOOLS TO AUTOMATED TESTING OF TELEMETRIC DATA PROCESSING SOFTWARE

The method of common automated testing scenarios design, classes structure for such scenarios, advantages and lacks are described in the article.

Automated testing, common scenarios, WindowTester Pro

УДК 681.3, 681.5

М. С. Куприянов, Ю. А. Шичкина

Схема укрупнения операций распараллеливаемого алгоритма

Рассматривается построение информационного графа по последовательному алгоритму и его преобразование к параллельной форме с достаточной шириной ярусов и с последующей реализацией на параллельной вычислительной среде. Показано также, что формализация данного процесса позволяет найти оптимальное решение задачи распараллеливания алгоритма с учетом таких параметров, как количество процессоров, время вычислений и плотность вычислений на единицу времени.

Параллельный алгоритм, информационный граф, ширина графа, список следования

Практически одновременно с появлением компьютера человечеству стало ясно, что созданные компьютеры не в состоянии решить за приемлемое время многие задачи. Несмотря на то, что производительность компьютеров росла экспоненциально начиная с 1945 г. и продолжает расти в настоящее время, потребность в увеличении скорости обработки данных превышает показатели этого роста.

Параллельное программирование, как и обычное программирование, опирается на алгоритмы и структуры данных. Неэффективно работающая программа – это прямые потери производительности компьютера, средств на его приобретение, усилия на освоение и т. п. Таких потерь хотелось бы избежать или, по крайней мере, их минимизировать. Для этого необходимы апробированные методика анализа алгоритмов и методы их оптимизации.

Чаще всего оптимизация параллельных алгоритмов рассматривается в условиях неограниченного параллелизма, которые предполагают использование идеализированной модели параллельной вычислительной системы и которых на практике не существует.

Эффективность параллельного алгоритма зависит от многих параметров. Одним из них является равномерная загрузка процессоров вычислительной системы. При этом важно, с одной стороны, сохранив высоту параллельного алгоритма, минимизировать его ширину. Эту проблему можно решить с помощью метода, представленного в [1]. Другим важным параметром является время выполнения процессов. Если предполагать, что время выполнения всех процессов, отраженных в информационном графе в виде отдельных вершин, одинаково, тогда ориентированный ациклический информационный граф является наиболее подходящим инструментом для построения оптимального по высоте и ширине параллельного алгоритма. Но на практике такие условия – крайняя редкость. Следовательно, из-за неравномерности выполнения отдельных процессов часть процессоров будет простаивать в ожидании результатов от своих предшественников. Поэтому сама собой напрашивается замена ориентирован-

ного ациклического информационного графа его взвешенным аналогом, в котором в качестве весов будет выступать время выполнения процессов.

Далее предлагается метод, позволяющий оптимизировать параллельный алгоритм по времени при сохранении или уменьшении числа вычислительных единиц.

Метод укрупнения операций алгоритма по взвешенному информационному графу. Временной список следования – совокупность элементов, полученных на основе матрицы следования по правилу: если на пересечении *i*-й строки и *j*-го столбца матрицы смежности стоит единица, то номера этих строки и столбца образуют в списке элемент $\left(v_i,\ v_j, t_j\right)$, где t_j – время выполнения операции v_j.

Список следования позволяет сэкономить память при хранении информации о взвешенном графе и увеличить скорость обработки данных при анализе алгоритма на существование простоев процессоров при его реализации.

Порядок выполнения операций следующий:

1. Рассчитать значение теоретической минимальной ширины *d* информационного графа [1].

2. Добавить в граф вершину. Все выходные вершины соединить с новой вершиной. Таким образом, в графе будет всего одна выходная (*n*+1)-я вершина.

3. Построить соответствующий информационному графу временной список следования $V_0 = \left(V_1,\ V_2, t\right)$.

4. Построить множество выходных вершин V_k.

5. Найти множество $V = V_2 - V_1$. Вершины, вошедшие в это множество, составят группу вершин M_i, принадлежащих одному i-му ярусу.

• Если число вершин множества $V = V_2 - V_1$ превышает *d*, то возможны следующие варианты:

а) во множестве M_i существует ровно *d* вершин с одинаковым временем выполнения. В данном случае эти *d* вершин составят множество M_i;

б) во множестве M_i существует больше чем *d* вершин с одинаковым временем выполнения. В этом случае любые *d* вершин с одинаковым временем выполнения составят множество M_i;

в) во множестве M_i существует менее *d* вершин с одинаковым временем выполнения. В этом случае множество M_i составят *d* вершин с минимальным временем выполнения.

Остальные вершины необходимо включить в следующую группу M_{i+1}. При повторном проходе этого шага в группу M_{i+1} будут добавлены вершины.

• Если число вершин множества $V = V_2 - V_1$ меньше числа *d*, то во множество M_i следует перенести вершины из множества V_k, удовлетворяющие правилу: $V_{kj} \cap V_1 = \varnothing$, т. е. вершины из V_k, отсутствующие во множестве V_1. Эти вершины могут быть взяты в произвольном порядке из множества: $V_k - V_1$.

6. Удалить из списка смежности все пары, конечная вершина которых (V_2) совпадает с одной из вершин множества V, и построить тем самым список V_i.

7. Если $i = 1$, то вернуться на шаг 5.

8. Во множестве M_{i-1} найти вершину с минимальным временем выполнения m_{i-1k}. Если все вершины имеют одинаковое время выполнения, то перейти к шагу 16.

9. Составить множества:

– $S_{m_{i-1k}M_i}$ – совокупность вершин из множества M_i, связанных с вершиной m_{i-1k} одним ребром и являющихся для данного ребра конечными вершинами;

– $S_{(M_{i-1}-m_{i-1k})M_i}$ – совокупность вершин из множества M_i, связанных с вершинами множества M_{i-1} (за исключением вершины m_{i-1k}) одним ребром и являющихся для данного ребра конечными вершинами.

Внимание! Если во множестве M_{i-1} есть укрупненные операции, то рассматривается всегда только последняя добавленная при укрупнении вершина.

10. Найти разность: $S = S_{m_{i-1k}M_i} - S_{(M_{i-1}-m_{i-1k})M_i}$.

• Если $S = \varnothing, S_{m_{i-1k}M_i} \neq \varnothing, S_{(M_{i-1}-m_{i-1k})M_i} \neq \varnothing$, то перейти к шагу 5.

• Если $S_{m_{i-1k}M_i} = \varnothing, S_{(M_{i-1}-m_{i-1k})M_i} = \varnothing$, то $S = M_i$.

• Если $S_{m_{i-1k}M_i} = \varnothing, S_{(M_{i-1}-m_{i-1k})M_i} \neq \varnothing$, то $S = M_i - S_{(M_{i-1}-m_{i-1k})M_i}$.

№	V_1	V_2	t
1	6	8	3
2	9	8	3
3	4	3	3
4	12	8	3
5	12	9	2
6	10	8	3
7	10	4	2
8	10	3	3
9	10	9	2
10	10	6	1
11	7	8	3
12	7	4	2
13	7	3	3
14	7	9	2
15	7	6	1
16	7	10	1
17	11	8	3
18	11	4	2
19	11	3	3
20	11	9	2
21	11	6	1
22	11	10	1

№	V_1	V_2	t
23	14	8	3
24	14	4	2
25	14	3	3
26	14	9	2
27	14	6	1
28	14	10	1
29	14	7	2
30	5	4	2
31	5	3	3
32	5	1	2
33	2	1	2
34	15	1	2
35	15	2	1
36	13	12	3
37	16	8	3
38	16	4	2
39	16	3	3
40	16	9	2
41	16	6	1
42	16	10	1
43	16	13	1
44	16	7	2

№	V_1	V_2	t
45	16	1	2
46	16	14	2
47	16	2	1
48	16	12	3
49	16	15	3
50	17	1	2
51	17	2	1
52	17	3	3
53	17	4	2
54	17	5	2
55	17	6	1
56	17	7	2
57	17	8	3
58	17	9	2
59	17	10	1
60	17	11	3
61	17	12	3
62	17	13	2
63	17	14	2
64	17	15	3
65	17	16	1

Рис. 1

11. Найти во множестве вершину с минимальным временем выполнения $s_{\min}$. Удалить эту вершину из множества M_i. Добавить эту вершину во множество M_{i-1}, укрупнив операцию m_{i-1k} слиянием ее с операцией $s_{\min}$:

$$m_{i-1k} = m_{i-1k} \cup s_{\min}, t_{i-1k} = t_{i-1k} + t_{s_{\min}}.$$

12. Если $M_i = \varnothing$, то удалить множество $M_i = \varnothing$ и сдвинуть счетчик множеств $i = i - 1$. Перейти к шагу 4. Если $M_i \neq \varnothing$, то перейти к шагу 9.

13. Если список не пустой, то вернуться на шаг 4.

14. Если в списке не осталось ни одной пары, работа окончена.

Общее время работы алгоритма составит $T = \sum_{i=1}^{k} t_{i,\max}$, где k – число групп; $t_{i,\max}$ – максимальное время среди операций в i-й группе.

Пример. Первоначальный список следования, соответствующий некоторому информационному графу, представлен на рис. 1.

Найдем разность множеств $V = V_2 - V_1 = \{1, 3, 8\}$. Эта группа вершин составит первый ярус информационного графа: $M_1 = \{1, 3, 8\}$ (рис. 2).

M_1	
i	t
1	2
3	3
8	3

Рис. 2

Теоретическая минимальная ширина графа равна 3, поэтому в данную группу пока не будем добавлять вершины из конечной группы V_k. Удалим из первоначального списка все пары, вторым элементом которых является одна из вершин группы M_1, и получим новый список. Вновь найдем разность множеств конечных и начальных вершин ребер графа $V = V_2 - V_1 = \{9, 4, 6, 2\}$ (рис. 3).

M_2	
j	t
9	2
4	2
6	1
2	1

Рис. 3

Во множестве M_1 найдем вершину с минимальным временем выполнения $m_{11}=1, t_1=2$. Составим множества S_{1M_2} (совокупность вершин из множества M_2, связанных с вершиной 1 одним ребром и являющихся для данного ребра конечными вершинами) и $S_{\{3,8\}M_2}$ (совокупность вершин из множества M_2, связанных с вершинами множества M_1, за исключением вершины 1, одним ребром и являющихся для данного ребра конечными вершинами). Найдем разность: $S = S_{1M_2} - S_{\{3,8\}M_2}$ (рис. 4).

S_{1M_2}	$S_{\{3,8\}M_2}$	$S = S_{1M_2} - S_{\{3,8\}M_2}$
4	4	2
6	9	
2	6	

Рис. 4

Во множестве S одна вершина 2. Удалим эту вершину из множества M_2 и добавим ее во множество M_1, укрупнив тем самым операцию 1 слиянием ее с операцией 2: $1' = \{1, 2\}$, $t_1 = t_1 + t_2 = 2 + 1 = 3$ (рис. 5).

Скорректированные группы				
M_1			M_2	
i	t		j	t
1.2	3		9	2
3	3		4	2
8	3		6	1

Рис. 5

В результате, первая группа выровнена по времени. На этом ее фиксируем. Дальнейшим изменениям данная группа не подлежит. Продолжая далее, получим в итоге 3 группы (рис. 6).

M_1			M_2			M_3	
i	t		j	t		b	t
1.2	3		9,12,13,14	8		16	1
3	3		4,10,5,11	8			
8	3		6,15,7	6			

Рис. 6

Общее время работы алгоритма

$$T = \sum_{i=1}^{k} t_{i,\max} = 3 + 8 + 1 = 12 .$$

Проиллюстрируем результаты с помощью временных диаграмм.

До применения метода оптимизации по времени с помощью метода оптимизации по ширине [1] были получены 6 групп операций для трех процессоров (рис. 7).

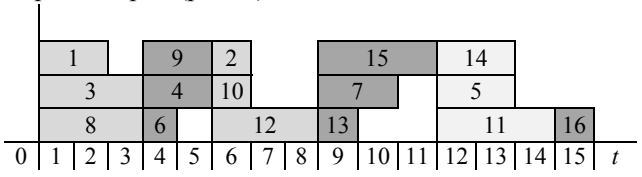

Рис. 7

Время работы алгоритма равно 15 ед.

После применения метода оптимизации по времени с сохранением минимальной теоретической ширины $d = 3$ было получено 3 группы (рис. 8).

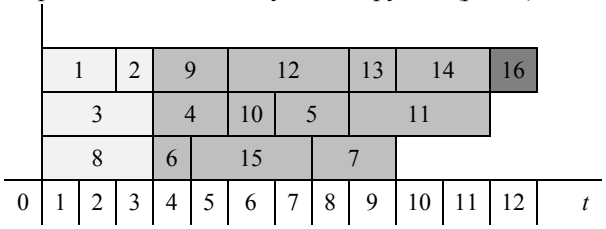

Рис. 8

Время работы алгоритма равно 12 ед.

Анализ трудоемкости показал, что применение временных списков следования более эффективно по сравнению с аналогичным методом, состоящим из двух частей [2] и основанным на матрице смежности. Метод оптимизации информационного графа по времени с применением временных списков следования позволяет значительно сократить время работы алгоритма с сохранением заданной ширины графа. При этом ширина графа может быть оговорена произвольным образом: либо на основании найденной теоретической минимальной ширины графа, либо в соответствии с особенностями вычислительной системы.

Следует заметить, что решение задачи минимизации алгоритма по времени вычисления может быть не единственным. Применение изложенного метода оптимизации информационного графа по времени с использованием временных списков следования позволяет свести время выполнения алгоритма к минимальному при заданной ширине и найти только одно из решений задачи минимизации. Так, в приведенном примере существует еще одно решение (рис. 9).

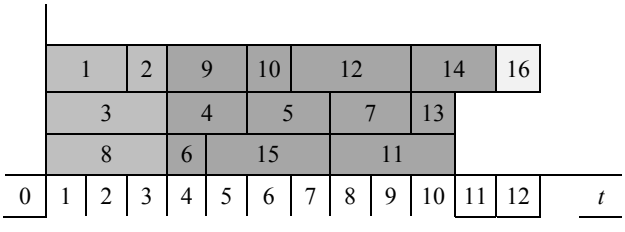

Рис. 9

Время работы алгоритма равно 12 ед.

Работа выполнена в СПбГЭТУ при финансовой поддержке Министерства образования и науки Российской Федерации (договор № 02.G25.31.0058 от 12.02.2013).

СПИСОК ЛИТЕРАТУРЫ

1. Шичкина Ю. А. Сокращение высоты информационного графа параллельного алгоритма // Науч.-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. 2009. № 3 (80). С. 148–152.

2. Шичкина Ю. А. Применение списков следования для оптимизации информационного графа по высоте // Системы. Методы. Технологии. 2011. № 1 (9). С. 68–77.

M. S. Kupriyanov, Yu. A. Shichkina

SCHEME OF OPERATIONS ENLARGING FOR PARALLELIZED ALGORITHMS

One of the solutions of the problem designing the effective numerical methods for parallel architecture computers is building the information graph by serial algorithm and transforming by list consecutions method to parallel form with sufficient edge width and implementation with parallel system. This article points out that it is possible not only to formalize this process but also to find out the optimal solution for parallelizing the algorithm with a glance to such parameters as a number of processor units, computation time and computation density per unit time.

Parallel algorithm, information graph, width of graph, list consecutions