УДК 681.3.06

МЕТОД ПОЛУЧЕНИЯ ОПТИМАЛЬНОГО БАЛАНСА НАГРУЗКИ ДЛЯ РАСПРЕДЕЛЕННОГО ЛОГИЧЕСКОГО МОДЕЛИРОВАНИЯ

В.А. Куркчи, Ю.В. Ладыженский
Кафедра ПМИ, ДонНТУ

Наведено опис алгоритму динамічного балансування навантаження, що базується на розв'язанні транспортній задачі. Запропоновано модифікацію алгоритму, що дозволяє гарантовано отримувати оптимальній баланс навантаження за рахунок незначного підвищення рівню міграції. Наведено результати експериментального порівняння алгоритмів.

Введение. В течение процесса логического моделирования при использовании вычислительной сети нагрузки на рабочие станции с течением времени могут значительно изменяться. Возникающая разбалансированность нагрузки приводит к неэффективному использованию оборудования. Поэтому важно периодически балансировать нагрузку [1].

С точки зрения распределенного логического моделирования среди ряда существующих методов балансировки нагрузки [2-6] особого внимания заслуживает метод, минимизирующий уровень миграции [7].

Чем меньше задач будет передано в процессе балансировки нагрузки, тем скорее можно будет продолжить процесс моделирования. Поэтому минимизация уровня миграции очень важна при балансировке нагрузки.

В нашей работе [8] был предложен алгоритм, позволяющий балансировать нагрузку с минимальным уровнем миграции. Алгоритм основан на решении транспортной задачи. Этот алгоритм позволяет в среднем на 23% сократить количество передаваемых подзадач [9], однако качество его решений может быть улучшено.

Целью данной работы является модификация указанного алгоритма для получения оптимальных решений.

Постановка задачи. Пусть P есть количество процессоров, а S множество всех подзадач, которые должны быть решены. Множество $C=\{C_1,C_2,...,C_P\}$ является размещением множества всех подзадач на процессоры. Каждая подмножество подзадач C_i обрабатывается на одном процессоре i. Подмножества C_i и C_j не пересекаются. Объединение всех

множеств C_i есть множество S. Обозначим через l_i=| C_i | нагрузку на один процессор и

$$l_{avg} = \frac{1}{P} \sum_{C_i \in C} l_i$$

среднюю нагрузку на процессор.

Рассмотрим граф $G(C,E)$, где $(C_i,C_j) \in E$, если возможно переместить подзадачи от процессора i к процесору j. Если есть такое множество подзадач C_i в C, что $l_i > l_{avg}$, одна или несколько подзадач $k \in C_i$ должны быть перемещены в другую часть графа подзадач. Часть графа подзадач, принимающая вершины может быть выбрана только среди множества смежных вершин графа $\{C_j | (C_i,C_j) \in E\}$.

Задача динамической балансировки нагрузки заключается в поиске нового размещения подзадач $C' = \{C'_1, C'_2, \ldots, C'_P\}$, которое минимизирует одновременно дисбаланс и уровень миграции. Дисбаланс можно найти по одной из следующих формул [5], [6], [3]:

$$M_1(C) = \max_{k=\overline{1,P}} |l_k| - \min_{m=\overline{1,P}} |l_m|,$$

$$M_2(C) = \frac{1}{l_{\max}} \sqrt{\frac{\sum_{i=1}^{P} (li - l_{avg})^2}{P-1}},$$

$$M_3(C) = \max_{i=\overline{1,P}} l_i,$$

где

$$l_i = | C_i |, \; l_{\max} = \max_{i=\overline{1,P}} l_i, \; l_{avg} = \frac{1}{P} \sum_{i=1}^{P} l_i .$$

Уровень миграции вычисляется следующим образом:

$$Q(C) = \frac{\sum_{k=1}^{P} (|C_k| - |C_k \setminus C'_k|) + |C'_k \setminus C_k|}{2}.$$

Алгоритм, основанный на решении транспортной задачи. Задачу балансировки нагрузки можно рассматривать как задачу определения числа подзадач, которые необходимо переместить с каждого процессора и указания целевых процессоров для каждой перемещаемой подзадачи.

Есть несколько процессоров с нагрузкой $l_i > l_{avg}$. Каждый из этих процессоров должен передать $\lfloor l_i - l_{avg} \rfloor$ подзадач. Также есть некоторое число процессоров с нагрузкой $l_i<l_{avg}$,, каждый из которых должен получить $\lfloor l_{avg} - l_i \rfloor$ подзадач. Если определить расстояние между двумя процессорами, описанная задача может быть решена как транспортная с помощью метода потенциалов. Метод потенциалов описан в [10].

Псевдокод алгоритма балансировки нагрузки, основанного на решении транспортной задачи, представлен на рис. 1. Этот алгоритм возвращает матрицу Δ размера $P \times P$, $\Delta[i,j]$ является количеством подзадач передаваемых от процессора i процессору j. Начальные значения $\Delta[i,j]$ должны быть равны нулю. На первом шаге алгоритм вычисляет среднюю нагрузку на процессор. Затем формируются множества O и I процессоров, которые должны соответственно передавать и получать подзадачи. На шаге 4 вызывается алгоритм Флойда, результатами работы которого являются матрица D длин путей матрица W, характеризующая последовательность вершин для каждого пути.

На шагах 6-8 формируются исходные данные для метода потенциалов: длины кратчайших путей между процессорами, передающими подзадачи, и принимающими их, заносятся в матрицу T. Затем на шаге 9 вызывается метод потенциалов, чтобы решить транспортную задачу. Он использует матрицу T длин путей между процессорами, , множество $O - \lfloor l_{avg} \rfloor$ количеств подзадач, которые требуется передать каждому процессору с избыточной нагрузкой, множество $\lfloor l_{avg} \rfloor - I$ количеств подзадач, которые может получить каждый процессор с недостатком нагрузки. Результатом метода потенциалов является матрица TS размера $|O| \times |I|$, каждый элемент $TS[k,m]$ есть количество подзадач, которое следует передать от процессора $O[k]$ процессору $I[m]$.

После этого, результаты метода потенциалов должны быть интерпретированы. Для каждой пары процессоров «передающий - принимающий» (шаги 10-11) необходимо восстановить кратчайший путь (шаги 12-16). Переменная c для восстановления пути инициализируется на шаге 12 и ссылается на текущий процессор пути. Количество подзадач, передаваемых от процессора k процессору m, добавляется к каждому ребру пути в графе процессоров, как это показано на рис. 1. В реализации

алгоритма это количество равно $TS[k,m]$, и оно прибавляется к элементу $\Delta[c,W[c,m]]$ возвращаемой матрицы, где $W[c,m]$ – следующий процессор пути [см. 11]. Эти операции выполняются на шагах 13-16, пока переменная c, представляющая текущий процессор, не станет равной процессору m.

1. $l_{avg} = \left\lceil \frac{1}{P} \sum_{i=1}^{P} C_i \right\rceil$
2. $O \leftarrow \{C_i \mid |C_i| > l_{avg}\}$
3. $I \leftarrow \{C_i \mid |C_i| < l_{avg}\}$
4. $\langle D, W \rangle \leftarrow Floyd(G(C, E_N))$
5. $T \leftarrow CreateMatrix(|O| \times |I|)$
6. *foreach* $k \in O$
7. *foreach* $m \in I$
8. $T[k,m] \leftarrow D[k,m]$

endfor

endfor

9. $TS \leftarrow SolveTransport(T, O - \lfloor l_{avg} \rfloor, \lfloor l_{avg} \rfloor - I)$
10. *foreach* $k \in O$
11. *foreach* $m \in I$
12. $c := vs$
13. *repeat*
14. $\Delta[c,W[c,m]] \leftarrow \Delta[c,W[c,m]] + TS[k,m]$
15. $c \leftarrow W[c,m];$
16. *until* $c=m;$

endfor

endfor

Рис. 1. Псевдокод алгоритма балансировки нагрузки, основанного на решении транспортной задачи.

Модификация алгоритма. Описанный алгоритм позволяет всегда получать наименьшую максимальную нагрузку на процессор $M_1(C)$. Это достигается за счет того, что вся нагрузка выше средней передается другим процессорам. При этом также минимизируется количество

передаваемых подзадач – оптимальность решения достигается за счет использования метода потенциалов.

Однако минимальная нагрузка на процессор может остаться неизменной. По этой причине показатели сбалансированности нагрузки $M_2(C)$ и $M_3(C)$ оказываются большими.

Рассмотрим причины появления таких решений на примере. Пусть нагрузка на 10 процессорах распределена следующим образом: 100, 104, 96, 96, 99, 100, 103, 101, 102, 96. Средняя нагрузка на процессор l_{avg}=99.7, поэтому нагрузка свыше $\lceil l_{avg} \rceil$=100 подзадач будет передана другим процессорам.

После применения описанного алгоритма нагрузка распределена следующим образом: 100, 100, 100, 100, 100, 100, 100, 100, 100, 97. Очевидно, что сделать нагрузку меньше 100 подзадач невозможно, но возможно распределить нагрузку следующим образом: семь процессоров по 100 подзадач и три по 99.

Так как 997 подзадач не могут равномерно распределиться на 10 процессорах – необходимо еще 3 задачи – оказывается, что избыточная нагрузка на все процессоры составляет 10 подзадач, а недостаток нагрузки – 13 подзадач. В таких случаях для решения задачи методом потенциалов вводятся фиктивный процессор с фиктивной лишней нагрузкой, которая удаляется при составлении плана миграции. В данном случае вводятся 3 фиктивные подзадачи.

Метод потенциалов работает таким образом, что фиктивные подзадачи попадают на один, реже на два, процессора. Исправить этот недостаток внесением изменений в алгоритм оказалось невозможным.

Однако можно использовать алгоритм повторно, чтобы получить сбалансированную нагрузку. Для этого применим повторно алгоритм, но примем за среднюю нагрузку $\lfloor l_{avg} \rfloor$=99. Это создаст избыток лишней нагрузки.

В примере после повторного применения распределение нагрузки будет следующим: 99, 99, 100, 100, 100, 100, 100, 100, 100, 99.

Таким образом, способ применения алгоритма балансировки нагрузки, основанного на транспортной задаче, позволяющий получить оптимальный баланс состоит из двух этапов:

1. Применить для решения задачи алгоритм, основанный на транспортной задаче (см. рис 1). На шаге 1 алгоритма принять

$$l_{avg} = \left\lceil \frac{1}{P} \sum_{i=1}^{P} C_i \right\rceil$$

2. Применить для решения задачи алгоритм, основанный на транспортной задаче (см. рис 1). На шаге 1 алгоритма принять

$$l_{avg} = \left\lfloor \frac{1}{P} \sum_{i=1}^{P} C_i \right\rfloor .$$

Экспериментальные исследования. Обе версии алгоритма были программно реализованы. Также программно реализован алгоритм из [7]. Эксперименты проводились на случайных графах задач с 10000 вершин. Задачи распределялись между 10 процессорами.

В таблице 1 представлены характеристики полученных решений $M_1(C)$, $M_2(C)$, $M_3(C)$, при этом Н – для алгоритма из [7], Т – для исходного алгоритма, ТВ – для модифицированного алгоритма.

Из таблицы видно, что характеристика сбалансированности $M_1(C)$ одинакова для разработанных алгоритмов, но лучше, чем у алгоритма из [9]. Характеристики $M_2(C)$ и $M_3(C)$ существенно лучше у модифицированного алгоритма. Кроме того, $M^{TB}{}_2(C)=1$ для всех входных графов. Это означает, что на всех тестовых задачах достигнуто максимально сбалансированное решение, так как в случаях, когда количество подзадач не делится нацело на количество процессоров, разница в нагрузке на процессоры не может быть устранена полностью.

В таблице 2 представлены уровни миграции для тех же тестовых задач. Уровень миграции разработанных алгоритмов ниже, чем у алгоритма из [7] на 24.7% для исходного алгоритма и на 22.5% для модифицированного алгоритма. Таким образом, для получения оптимального баланса нагрузки па процессоры модифицированный алгоритм перемещает на 2,2% больше подзадач, чем исходный алгоритм.

Выводы. Проведен анализ недостатков работы алгоритма динамической балансировки нагрузки, основанного на решении транспортной задачи.

Разработана модификация алгоритма, позволяющая получать полностью сбалансированные решения.

Экспериментальные результаты показали, что решения, получаемые при помощи модификации алгоритма, действительно полностью сбалансированы, при этом уровень миграции по сравнению с исходным алгоритмом практически не изменяется.

Таблица 1- Характеристики полученных решений

$M^{H}_{1}(C)$	$M^{T}_{1}(C)$	$M^{TB}_{1}(C)$		$M^{H}_{2}(C)$	$M^{T}_{2}(C)$	$M^{TB}_{2}(C)$		$M^{H}_{3}(C)$	$M^{T}_{3}(C)$	$M^{TB}_{3}(C)$
502	501	501		4	19	1		0,002312	0,008265	0,000435
515	514	514		4	16	1		0,002854	0,006784	0,000778
507	506	506		4	16	1		0,001933	0,006892	0,000791
509	508	508		3	5	1		0,001506	0,002145	0,000852
486	484	484		5	8	1		0,003468	0,003942	0,001012
508	507	507		4	14	1		0,001978	0,006018	0,000904
492	489	489		5	3	1		0,002593	0,001337	0,00073
509	507	507		5	16	1		0,003204	0,006878	0,000789
509	507	507		5	9	1		0,002811	0,003869	0,000981
505	503	503		4	1	1		0,002299	0,000433	0,000433
501	500	500		4	11	1		0,002685	0,004903	0,000917
500	500	500		2	10	1		0,001183	0,004359	0,001
492	491	491		3	17	1		0,001849	0,007546	0,000727
476	474	474		5	1	1		0,002856	0,00046	0,00046
505	504	504		3	13	1		0,001907	0,005622	0,000946
509	507	507		3	2	1		0,002143	0,00086	0,000592
510	510	510		3	18	1		0,001743	0,007692	0,000588
520	519	519		3	10	1		0,001654	0,004239	0,000944
497	494	494		6	15	1		0,003237	0,006618	0,000877
501	499	499		4	16	1		0,001851	0,006988	0,000802

Таблица 2 – уровень миграции для полученных решений

№	Q^{H}	Q^{T}	Q^{TB}		№	Q^{H}	Q^{T}	Q^{TB}
1	510	386	404		11	536	419	431
2	485	353	368		12	441	324	333
3	558	414	429		13	553	419	435
4	385	273	277		14	424	308	308
5	427	319	329		15	472	347	359
6	514	383	396		16	509	382	383
7	503	417	419		17	561	444	461
8	504	370	385		18	514	396	406
9	457	359	367		19	436	300	314
10	517	391	391		20	478	369	384

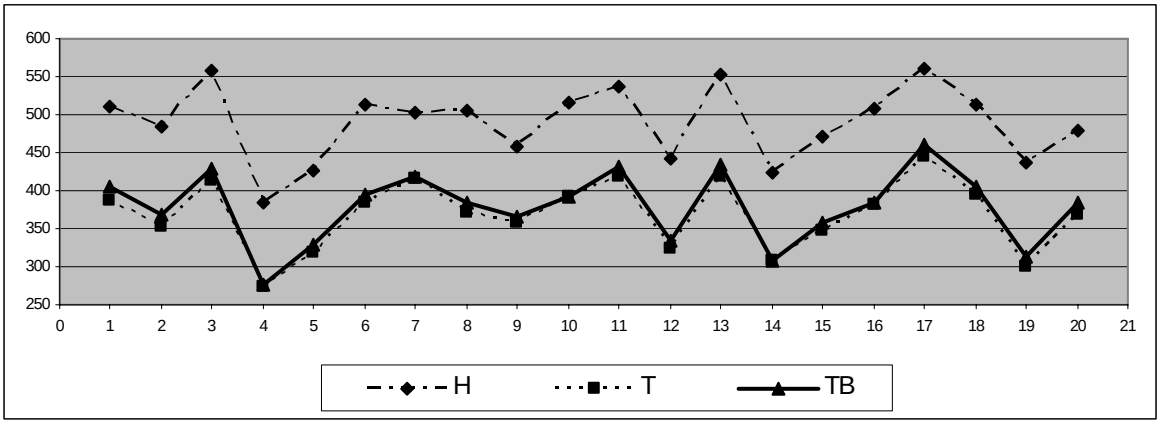

Рисунок 2 – Уровень миграции для каждой тестовой задачи

Литература

1. Ю. Ладыженский, В. Куркчи, Балансировка нагрузки в распределенном логическом моделировании//ІНТЕРНЕТ-ОСВІТА-НАУКА-2006, п'ята міжнародна конференція ІОН-2006, 10-14 жовтня, 2006, збірник матеріалів конференції. Том 2. – Вінниця: УНІВЕРСУМ-Вінниця, 2006, – 647-653.

2. K. Devine and al., A new challenges in dynamic load balancing.//Applied Numerical Mathematics, v 52, 2005, pp 133-152.

3. A. Pothen, Graph partitioning algorithms with applications to scientific computing.//Parallel Numerical Algorithms, Kluwer Academic Press, 1996.

4. G. Karypis, V. Kumar, Analysis of multilevel graph partitioning. Tech. Report 95-035, Computer Science Department, University of Minnesota, 1995.

5. H. Avril, C. Tropper, Clustered time warp and logic simulation//9th Workshop on Parallel and Distributed Simulation (PADS'95), 1995, pages 112-119.

6. H. Avril, C. Tropper, The Dynamic Load Balancing of Clustered Time Warp for Logic Simulations//Workshop on Parallel and Distributed Simulation, 1996, pages 20-27.

7. Y.F. Hu, R.J. Blake, An optimal dynamic load balancing algorithm. Preprint DL-P-95-011, Daresbury Laboratory, Warrington, WA44AD, UK. (To be published in Concurrency: Practice & Experience), 1995.

8. Y.V. Ladyzhensky, V.V. Kourktchi, Transport-Problem-Based algorithm for dynamic load balancing in logic simulation.//Information Systems Technology and its Applications: Proceedings of 6th international conference ISTA'2007, may 23-25, 2007, Kharkiv, Ukraine. – Lecture Notes in Informatics, Gesellschaft fur Informatik, Bonn 2007.

9. Ю.В. Ладыженский, В.А. Куркчи. Исследование свойств алгоритмов для динамической балансировки нагрузки в распределенном логическом моделировании.//Вісник Національного технічного університету „Харківський політехнічний інститут": Збірник наукових праць. Тематичний випуск „Системний аналіз, управління та інформаційні технології". – Харків: НТУ „ХПІ". –2007. –№6. – 24-34.

10. Венцель Е.С. "Исследование операций: задачи, принципы, методология" -М.: "Наука",1980. - 532с.

11. Дискретная математика для программистов / Ф.А. Новиков – СпБ: Питер, 2001. – 304с.: ил.

Работа поступила в редакцию 08.06.2007.