Гончар Д.Р.

Параллельная реализация решения минимаксной задачи составления расписания методом ветвей и границ

Аннотация: Рассматривается минимаксная задача построения расписания наименьшей длины без прерываний для многопроцессорной системы. Для решения данной задачи предложен параллельный алгоритм на основе метода ветвей и границ. Исследовано реальное ускорение расчётов при увеличении числа используемых вычислительных ядер.

Ключевые слова: многопроцессорная система, работы без прерываний, расписание наименьшей длины

Одним из очевидных способов ускорения расчётов при решении задач по построению оптимальных расписаний является разработка параллельных реализаций используемых алгоритмов планирования. В данной работе приведено как описание самого алгоритма на основе метода ветвей и границ, так и итоги расчётов с использованием разного числа процессоров на вычислительном комплексе Межведомственного суперкомпьютерного центра РАН (МСЦ РАН).

Постановка задачи. Пусть имеется множество работ $N = \{1, 2, \ldots, n\}$, которое необходимо выполнить с помощью m процессоров, составляющих вычислительную систему для их обработки. Длительность выполнения работы i на процессоре j равно t_{ij} ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$). В каждый момент времени каждая работа может выполняться не более чем одним процессором, а каждый процессор может выполнять не более одной работы. Переключения с одного процессора на другой и прерывания при выполнении работ не допускаются.

Расписание выполнения работ N определим как разбиение множества N на m непересекающихся подмножеств $N_1, N_2, \ldots, N_m$ ($N = \bigcup_{j=1}^{m} N_j$; $N_{j_1} \bigcap N_{j_2} = \varnothing$ при $j_1 \neq j_2$). Работы из множества N_j приписываются процессору j и выполняются на нем одна за другой в произвольном порядке. Под загруженностью процессора j ($j = 1,$

2, …, m) будем понимать величину $Q_j = \sum_{i \in N_j} t_{ij}$, а $\max_{j=1, 2, \ldots, m} Q_j$ – это длина расписания. Задача заключается в построении оптимального по быстродействию расписания, т.е. расписания минимальной длины.

Подобные задачи широко освещены в литературе. При их решении применяются, например, такие методы, как случайный и исчерпывающий поиск, методы математического программирования [1], метод ветвей и границ [2, 6], муравьиные алгоритмы, поиск с запретами, вероятностные алгоритмы, генетические алгоритмы [3], метод имитации отжига, различные эвристические алгоритмы [4, 5], алгоритмы агрегирования и др.

Метод ветвей и границ. Для решения вышеприведенной задачи предлагается метод ветвей и границ, основанный на результатах работы [2]. Этот метод подразумевает структуру поиска оптимального решения в виде дерева (что выполняется для нашей задачи), последовательное разбиение исходного множества решений на подмножества (ветви дерева решений) и применение оценочных процедур для определения перспективности подробного исследования очередной ветви дерева решений. На каждом последующем шаге новые подмножества образуются в результате разбиения некоторых подмножеств, полученных на предыдущих шагах, пока для подмножеств, соответствующих концевым вершинам дерева, решение задачи уже не требует разбиения.

В итоге указанного разбиения мы получаем множество подзадач, которые могут обрабатываться независимо (совмещено, одновременно по времени).

Дерево решений. Опишем множество всех расписаний (их число равно m^n) в виде дерева решений. Корень дерева находится на нулевом уровне и соответствует множеству всех расписаний. На первом уровне находится m вершин, каждая из которых соответствует множеству всех расписаний, в которых первая работа назначена на определенный процессор. На n-м уровне дерева расписаний находится m^n листьев, каждый из которых соответствует некоторому расписанию выполнения множества работ N.

Пусть x_k – некоторый узел уровня k дерева расписаний, $R(x_k)$ – множество всех расписаний, соответствующих этому узлу (т.е. множество расписаний, в которых работы 1, 2, …, k назначены на

определенные процессоры), x_{k+1}^{j} – узел уровня $k+1$ ($k < n$), связанный с узлом x_k ребром, соответствующим процессору j. Наша задача – вычисление нижней и верхней оценок минимальной длины расписания на множестве $R(x_k)$. Имея эти оценки, можно применить стандартную схему метода ветвей и границ [6] (например, одностороннего или фронтального ветвления).

Вычисление нижней оценки. Пусть T_j (j = 1, …, m) – загруженность процессора j после назначения первых k работ (т.е. T_j – это суммарная длительность работ из числа 1, 2, …, k, назначенных на процессор j). Нижнюю оценку $L(x_k)$ минимальной длины расписания на множестве $R(x_k)$ будем вычислять следующим образом:

$$L(x_k) = \max(L_1(x_k), L_2(x_k), L_3(x_k)),$$

где $L_1(x_k)$, $L_2(x_k)$, $L_3(x_k)$ – это нижние оценки, вычисленные тремя различными способами.

Величина $L_1(x_k)$ вычисляется как следующий максимум: $L_1(x_k) = \max\limits_{j=1,2,\ldots,m} T_j$. При хранении величины T_1, T_2, …, T_m в виде обычного массива сложность вычисления $L_1(x_k)$ составляет $\theta(m)$.

Величина $L_2(x_k)$ вычисляется как следующий максимум:

$$L_2(x_k) = \max_{i=k+1,\ldots,n} \min_{j=1,\ldots,m} (T_j + t_{ij})$$

При использовании для этого двумерного массива A с элементами $a_{ij} = T_j + t_{ij}$, $i = k+1$, …, n; j = 1, 2, …, m сложность вычисления величины $L_2(x_k)$ составляет $\theta(mn)$.

Величина $L_3(x_k)$ вычисляется по формуле

$$L_3(x_k) = \frac{1}{m}\left(\sum_{j=1}^{m} T_j + \sum_{i=k+1}^{n} \min_{j=1,\ldots,m} t_{ij}\right).$$

Величину $\min\limits_{j=1,\ldots,m} t_{ij}$ вычислим для всех i = 1, 2, …, n сразу до начала вычисления нижних оценок. Тогда сложность вычисления величины $L_3(x_k)$ составляет $O(n + m)$. Перейдем в дереве расписаний от узла x_k к узлу $x_{k+1}^{j_0}$, $k < n$ (т.е. будем считать, что работа $k+1$ назначена на процессор j_0).

Тогда

$$L_3(x_{k+1}^{j_0}) = \frac{1}{m}\left(\sum_{j=1}^{m} T_j + t_{k+1,j_0} + \sum_{i=k+1}^{n} \min_{j=1,\dots m} t_{ij}\right).$$

Вычислим разность

$$L_3(x_{k+1}^{j_0}) - L_3(x_k) = \frac{1}{m}\left(t_{k+1,j_0} - \min_{j=1,\dots m} t_{k+1,j}\right).$$

Таким образом,

$$L_3(x_{k+1}^{j_0}) = L_3(x_k) + \frac{1}{m}\left(t_{k+1,j_0} - \min_{j=1,\dots m} t_{k+1,j}\right)$$

и с помощью данного рекуррентного соотношения, используя $L_3(x_k)$, величина $L_3(x_{k+1}^{j_0})$ вычисляется за время $O(1)$.

Вычисление верхней оценки. В качестве верхней оценки $H(x_k)$ минимальной длины расписания на множестве $R(x_k)$ возьмем длину расписания, в котором работы 1, 2, …, k в соответствии с вершиной x_k дерева расписаний распределены на процессоры, а работы k+1, …, n распределяются по следующему «жадному» алгоритму. Пусть уже распределены работы 1, 2, …, p ($k \le p < n$), T_j – загруженность процессора j (j = 1, 2, …, m) и $\min(T_1 + t_{p+1,1}, \dots, T_m + t_{p+1,m}) = T_{j_0} + t_{p+1,j_0}$. Тогда работа p+1 назначается на процессор j_0.

Указанные действия повторяются для $p = k, k + 1, \dots, n - 1$. Сложность процедуры вычисления величины $H(x_k)$ составляет $O(mn)$.

Ветвление. При применении метода ветвей и границ происходит последовательное разбиение множества допустимых решений на подмножества: на каждом последующем шаге новые подмножества образуются в итоге разбиения некоторых подмножеств, полученных на предыдущих шагах. Так строится уже упоминавшееся выше дерево решения исходной задачи. Такое разбиение продолжается до тех пор, пока для подмножеств, соответствующих концевым вершинам дерева, решение задачи уже не требует разбиения.

В итоге разбиения начальная задача распадается на ряд подзадач, которые могут решаться в значительной степени независимо друг от друга. Однако вследствие того, что дерево решения после разбиения может быть плохо уравновешенным,

целесообразно поддерживать определенные связи (зависимости) между полученными подзадачами. При этом необходимо учитывать величину накладных расходов, возникающих для поддержания указанных связей, иначе может произойти потеря эффективности при распараллеливании.

Для преодоления перечисленных причин снижения успешности распараллеливания решения задачи применяются методы оптимизации загрузки процессов, минимизации обменов данными, а также распределения обменов по вычислительному пространству [6, 7].

Итоги расчётов. В практическом плане нам интересно, насколько в действительности ускоряются расчёты по решению задачи при её распараллеливании на данном числе процессоров.

Таблица 1 – Длительность выполнения программы при m = 3 и различной степени параллельности

n (число заданий)	Число процессоров	Время счёта	Ускорение
16	1	0,86	
16	8	0,247	3,48
16	16	0,072	11,94
20	1	52,09	
20	8	6,23	8,36
20	64	0,9	57,88
22	1	409,68	
22	8	39,78	10,3
22	64	6,0	68,28
23	1	1177,46	
23	8	134,82	8,73
23	64	15,59	75,53
24	1	3392,75	
24	8	387,6	8,75
24	64	46,24	73,57
25	8	382,09	
25	64	128,39	3,02
26	8	2601,67	
26	64	376,53	6,91
27	8	8941,03	
27	64	1047,61	8,53

Отметим, что на длительность выполнения программы на МСЦ РАН достаточно заметное влияние оказывает общая нагрузка на систему, поэтому становятся возможными, на первый взгляд, такие парадоксальные значения, как ускорение счёта более чем в 8 раз при распараллеливании на 8 процессорах (вычислительных ядрах).

Литература:

1. *Алексеев О.Г.* Комплексное применение методов дискретной оптимизации. – М.: Наука, 1987. – 250 с.

2. *Фуругян М.Г.* Некоторые алгоритмы решения минимаксной задачи составления многопроцессорного расписания//Известия РАН, ТиСУ. – 2014. – № 2. – С. 50-56.

3. *Костенко В.А., Смелянский Р.Л., Трекин А.Г.* Синтез структур вычислительных систем реального времени с использованием генетических алгоритмов//Программирование. – 2000. – № 5. – С. 63-72.

1. *Brucker P.* Scheduling Algorithms. – Heidelberg: Springer, 2007. – 371 p.

4. *Гончар Д.Р.* Параллельная реализация мультиоценочного алгоритма составления многопроцессорного расписания без прерываний//Некоторые алгоритмы планирования вычислений и методы многокритериальной оптимизации для многопроцессорных систем. – М.: ВЦ РАН, 2014. – С. 21-31.

5. *Посыпкин М.А., Сигал И.Х., Галимьянова Н.Н.* Алгоритмы параллельных вычислений для решения некоторых классов задач дискретной оптимизации. – М.: ВЦ РАН, 2005. – 43 с.

6. *Посыпкин М.А., Сигал И.Х., Галимьянова Н.Н.* Параллельные алгоритмы в задачах дискретной оптимизации: вычислительные модели, библиотека, результаты экспериментов. – М.: ВЦ РАН, 2006. – 50 с.

Кулида Е.Л., Лебедев В.Г.

Исследование алгоритмов оптимизации очередности и времен посадок воздушных судов

Аннотация: Доклад посвящен решению задачи оптимизации очередности и времен посадок воздушных судов. Описан подход и средства исследования алгоритмов