УДК 681.324.006.3

Оценка множества работ при решении задач распараллеливания

© Н.Б. Толпинская

МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

В статье содержится описание подходов к оценке множества работ, которые выполняются в определенный момент времени на многопроцессорных вычислительных машинах, и объемов работ на заданном отрезке времени. Эти оценки позволяют более эффективно планировать распределение работ по процессорам при статическом решении задач распараллеливания.

***Ключевые слова*:** *распараллеливание, планирование, загрузка, работа, оператор.*

При решении задач распараллеливания, в особенности при статическом их решении, требуется получить оптимальный или близкий к нему план выполнения множества взаимосвязанных работ. Планирование параллельной реализации таких работ имеет целью максимальное снижение разницы пиковой и реальной производительности многопроцессорной вычислительной системы путем уменьшения простоев решающих устройств. Особенно высокие требования к производительности вычислительных систем предъявляются при их использовании в сложных системах управления в реальном времени. В этом случае при планировании параллельного вычислительного процесса обязательно учитываются оценки времени решения отдельных задач и производится согласование их с циклическим режимом и частотой выдачи управляющей информации. Временные диаграммы решения задач оцениваются во всех возможных ситуациях управления, и вычислительная система выбирается по самой напряженной ситуации.

Для представления множества работ при решении задач распараллеливания может быть использован информационный граф $G = (X, P, \Gamma)$ [1, 2], который отражает частичную упорядоченность операторов с учетом их информационной и логической зависимости. Здесь X — множество вершин, P — множество весов вершин, Γ — множество дуг. Вершинами графа являются подзадачи или процессы (операторы). Дуги графа отражают наличие информационных или логических зависимостей между операторами. Вес t_i каждой i-ой вершины ($i = 1, ..., m$) определяет время выполнения соответствующей подзадачи, процесса или оператора. Вес может быть скалярной величиной, если многопроцессорная вычислительная система является однородной или векторной, если вычислительная система неодно-

родная. В последнем случае вес вершины — это множество времен выполнения оператора на процессорах соответствующего типа.

При выполнении распараллеливания специально оговаривается логическая структура задач. В частности, будем считать, что если алгоритм содержит циклы, то либо для рабочей части цикла вопрос распараллеливания решается отдельно, либо цикл должен быть развернут. Кроме того, можно предположить отсутствие в схеме алгоритма логических операторов, так как статическое распараллеливание чаще всего выполняется для множества крупных взаимосвязанных задач, состав которых не меняется в течение длительного периода времени. Тогда схема алгоритма, где эти задачи фигурируют как отдельные операторы, может иметь малое количество ветвей или вообще одну ветвь. К тому же, при поиске точных планов параллельной реализации операторов необходимо оценить все действительные варианты совместного выполнения операторов, т. е. предусмотреть все значения логических переменных.

При указанных выше ограничениях проблема распараллеливания сводится к одной из двух задач:

1) для заданного комплекса информационно взаимосвязанных работ, представленного графом $G = (X, P, \Gamma)$, и для заданного ограничения на время выполнения этих работ T ($T \geq T_{кр}$, где $T_{кр}$ — критический путь в графе G) выбрать комплектацию вычислительной системы (в случае однородной системы задача сводится к поиску минимального количества решающих устройств);

2) найти план выполнения заданного комплекса взаимосвязанных работ за минимальное время на заданной вычислительной системе.

Решение первой задачи для однородной вычислительной системы и множества работ, заданных графом $G = (X, P, \Gamma)$, сводится к распределению множества взаимно независимых операторов по процессорам и закреплению их во времени. Взаимная независимость операторов определяется, исходя из информационного графа G с учетом задающих связей, определяющих передачу данных между операторами, и транзитивных связей. Транзитивные связи введены направленно между всеми парами операторов, не объединенными задающими связями и входящими в один путь в графе G. Транзитивные связи полностью определяются задающими связями.

Множество взаимно независимых операторов (ВНО) включает группу операторов, которые потенциально можно выполнять одновременно, т.е. параллельно. Однако временные характеристики части операторов, входящих в одно множество ВНО, могут не допустить параллельного их выполнения. Кроме того, разные множества ВНО могут иметь непустое пересечение. Значит, возникает вопрос: с операторами какого множества ВНО эффективнее выполнять операторы,

входящие в непустое пересечение нескольких множеств ВНО? Поэтому выделение операторов, которые могут выполняться параллельно, и дальнейшее планирование их реализации необходимо выполнять с учетом закрепления операторов на оси времени.

Этот процесс можно осуществить, определив момент окончания выполнения i-го оператора — τ_i. С учетом информационной зависимости между операторами и ограничения на время выполнения всего алгоритма T могут быть определены ранний τ_{1i} и поздний $\tau_{2i}(T)$ сроки окончания исполнения оператора. Ранний срок окончания выполнения оператора (если началом отсчета является момент начала выполнения операторов) не может быть меньше максимальной из длин всех путей, заканчивающихся вершиной, соответствующей данному оператору. Поздний срок не может превышать разности между значением T и максимальной из длин путей, в первую из вершин которых входит дуга, выходящая из вершины, соответствующей данному оператору. При параллельной реализации алгоритма исходной задачи за ограниченное время T любой оператор должен быть закреплен на оси времени так, чтобы $\tau_{1i} \le \tau_i \le \tau_{2i}(T)$. Если мощность некоторого множества операторов ВНО, параллельное выполнение операторов которого предполагается реализовать, не превышает ресурса вычислительной системы — множества решающих устройств, то можно положить $\tau_i = \tau_{1i}$ для каждого оператора этого множества ВНО. Однако, если мощность множества операторов ВНО больше множества решающих устройств, то необходимо построить эффективный план параллельной реализации операторов соответствующего множества ВНО.

В целях планирования вычислительного процесса в рассмотрение вводятся [1] функция плотности загрузки $F(\tau_1, \ldots, \tau_m, t)$, функция $\Phi(\tau_1, \ldots, \tau_m, \theta_1, \theta_2)$ загрузки отрезка времени $[\theta_1, \theta_2]$, принадлежащего отрезку $[0, \ldots, T]$, и функция минимальной загрузки отрезка $[\theta_1, \theta_2]$ — $\varphi^{(T)}(\theta_1, \theta_2)$. Однако следует заметить, что определения функций плотности загрузки и функции минимальной загрузки следует уточнить.

Функция плотности загрузки вычисляется как сумма значений функции $f(\tau_i, t)$ для всех операторов $i = 1, \ldots, m$:

$$\overset{m}{\underset{i=1}{F}}\left(\tau_1,\ldots,\tau_m,t\right) = \sum f\left(\tau_i,t\right), \qquad (1)$$

где функция $f(\tau_i, t)$ отражает факт выполнения i-го оператора в момент времени t. Функция $f(\tau_i, t)$ должна возвращать значение 1, если i-й оператор работает в момент t, и 0, если он не работает в этот момент. Как указано в [1], $f(\tau_i, t) = 1$ при $\tau_i - t_i \le t \le \tau_i$. Но если момент t

совпадает с моментом окончания работы i-го оператора τ_i и одновременно начинает работать следующий за ним оператор, т. е. $\tau_j - t_i = t(i \neq j)$, то по формуле (1) в сумме по этим двум операторам будет получено значение 2. Это означает, что в момент t должны одновременно, т. е. параллельно, выполняться два оператора — процесса, а это не так. Поэтому для функции $f(\tau_i, t)$ справедлива формула:

$$f(\tau_i, t) = 1 \text{ при } \tau_i - t_i \le t < \tau_i,$$
$$f(\tau_i, t) = 0 \text{ — в противном случае.} \qquad (2)$$

По такой формуле в момент t будет учитываться только одна работа, а именно, та, которая завершает свое выполнение в момент t. Для того же оператора, который еще не начал свое выполнение и не загрузил работами по своему исполнению вычислительную систему, функция $f(\tau_i, t)$ будет возвращать 0, а значит, он не будет учитываться и при вычислении $F(\tau_1, \ldots, \tau_m, t)$ — функции плотности загрузки.

Функция минимальной загрузки отрезка времени $[\theta_1, \theta_2]$ — $\varphi^{(T)}(\theta_1, \theta_2)$ является основным средством оценки результатов планирования при решении задач распараллеливания. Она позволяет оценить минимальный объем работ по исполнению всего алгоритма, который при всех возможных способах закрепления операторов на оси времени обязательно необходимо выполнить на отрезке времени $[\theta_1, \theta_2]$ для того, чтобы обеспечить выполнение всего алгоритма за время, не превышающее заданное время T. При ее расчете для данного отрезка времени $[\theta_1, \theta_2]$ по всем операторам проводится анализ наилучшего их расположения относительно данного отрезка. Наилучшим является такое положение i-го оператора относительно отрезка времени $[\theta_1, \theta_2]$, при котором он наименьшим образом загружает отрезок времени $[\theta_1, \theta_2]$ работами по своему исполнению. Допустим не единственный вариант закрепления i-го оператора на оси времени. Однако наилучшее его расположение может быть выбрано из ограниченного числа вариантов, которые определяются двумя условиями: время окончания выполнения оператора равно раннему сроку окончания его выполнения $\tau_i = \tau_{1i}$, или время окончания выполнения оператора равно позднему сроку окончания его выполнения $\tau_i = \tau_{2i}(T)$. В первом случае объем работ по исполнению этого оператора, который обязательно должен быть выполнен на отрезке $[\theta_1, \theta_2]$, равен $(\tau_{1i} - \theta_1)$ (при условии, что $\tau_{1i} - t_i < \theta_1$). Во втором случае объем работ по исполнению этого i-го оператора, который обязательно необходимо выполнить на отрезке $[\theta_1, \theta_2]$, равен

$(\theta_2 - (\tau_{2i}(T) - t_i))$ (при условии, что $\tau_{2i}(T) - t_i < \theta_2$). Но, кроме указанных двух случаев, возможны еще два:

1) $\tau_{1i} \geq \theta_2$ и одновременно $\tau_{2i}(T) - t_i \leq \theta_1$, тогда объем работ по исполнению i-го оператора, который обязательно необходимо выполнить на отрезке $[\theta_1, \theta_2]$, равен $(\theta_2 - \theta_1)$;
2) $\theta_1 \leq \tau_{1i} - t_i \leq \tau_{2i}(T) \leq \theta_2$, тогда объем работ по исполнению i-го оператора, который обязательно необходимо выполнить на отрезке $[\theta_1, \theta_2]$, равен t_i.

Последний случай, при котором объем работ по исполнению i-го оператора равен t_i, в [1] не учтен при вычислении функции $\varphi^{(T)}(\theta_1, \theta_2)$, но он может иметь место, а значит, должен быть учтен. Тогда условный минимальный объем работ по исполнению i-го оператора на отрезке $[\theta_1, \theta_2]$ может быть вычислен по формуле

$$\varphi_i^{(T)}(\theta_1, \theta_2) = \min\left\{\xi(\tau_{1i} - \theta_1), \xi(\theta_2 - \tau_{2i}(t) + t_i), \theta_2 - \theta_1, t_i\right\}, \quad (3)$$

где $\xi(x) = x$ при $x \geq 0$; $\xi(x) = 0$ при $x < 0$.

Таким образом, общий объем работ по исполнению всего алгоритма, который необходимо выполнить на отрезке $[\theta_1, \theta_2]$, т.е. значение функции минимальной загрузки отрезка $[\theta_1, \theta_2]$ — $\varphi^{(T)}(\theta_1, \theta_2)$, можно вычислить по формуле

$$\varphi^{(T)}(\theta_1, \theta_2) = \sum_{i=1}^{m}\left(\min\left\{\xi(\tau_{1i} - \theta_1), \xi(\theta_2 - \tau_{2i}(T) + t_i), \theta_2 - \theta_1, t_i\right\}\right). \quad (4)$$

ЛИТЕРАТУРА

[1] Барский А.Б. *Параллельные процессы в вычислительных системах. Планирование и организация*. Москва, Радио и связь, 1990, 256 с.

[2] Миренков Н.Н. *Параллельное программирование для многомодульных вычислительных систем*. Москва, Радио и связь, 1989, 320 с.

Статья поступила в редакцию 10.06.2013

Ссылку на эту статью просим оформлять следующим образом:

Толпинская Н.Б. Оценка множества работ при решении задач распараллеливания. *Инженерный журнал: наука и инновации*, 2013, вып. 6. URL: http://engjournal.ru/catalog/it/hidden/778.html

Толпинская Наталья Борисовна родилась в 1953 г. Канд. техн. наук, доцент кафедры «Программное обеспечение ЭВМ и информационные технологии» МГТУ им. Н.Э. Баумана. Автор 19 печатных работ. Область научных интересов: параллельное программирование (планирование и алгоритмы), функциональное программирование и логическое программирование. e-mail: nbtol@mail.ru