TECHNISCHE UNIVERSITÄT CHEMNITZ

Ulrich Elsner

Graph Partitioning

(Разбиение графов)

A survey

(обзор)

перевел Шепель А.И.

студент ДонНТУ специальности СП

май 2006

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Preprint SFB393/97-27, Dec. 1997

Введение в проблему

1.1 Введение

Несмотря на то, что производительность классических компьютеров (с архитектурой фон Нойманна) растет чрезвычайно быстро, всегда существует потребность в большем чем, текущие технологии могут дать. На данный момент возможно поместить необходимое количество транзисторов на чипе и пока это число продолжает увеличиваться из года в год, но определенные физические барьеры уже возникают на горизонте, которые не могут быть превзойдены технологией, известной сегодня.

И хотя поиск новых методов и технологий, которые могут отодвинуть эти барьеры дальше или же вообще сломать их, остается важной задачей, вопрос “Что можно сделать сейчас, чтобы получить большую производительность?” является также актуальным.

Человечество до нашей эры использовало коллективный труд там, где отдельно взятый человек не способен был справиться с поставленной задачей (начиная коллективной охотой и заканчивая строительством пирамид). Та же самая идея может быть использована в применении к компьютерам: параллельное вычисление. Придуманная приблизительно 15 лет назад или же, точнее сказать, заимствованная идея параллелизма, привела к тому, что параллельные компьютеры достигают производительности, которая является невозможной или дорогостоящей на однопроцессорной машине. Поэтому параллельные компьютеры той или иной архитектуры стали в большей степени популярными в научно-техническом сообществе.

Также как и в командной работе, которая не всегда является достаточно эффективной (10 человек не могут вырыть скважину в 10 раз быстрее одного), а иногда и создает новые проблемы, не возникающие при работе по отдельности (как скоординировать 10000 рабов, задействованных в строительстве пирамиды), трудности возникают и на параллельных машинах, неизвестные программистам в однопроцессорной системе.

Во-первых, не все части задачи могут быть решены параллелено, некоторые должны выполняться последовательно. Это является ограничителем суммарно возможного выигрыша в скорости (закон [1] Ахмдала) для указанной задачи. Довольно часто последовательная часть работы не зависит от размера решаемой задачи и таким образом для большей проблемы (при адекватном распределении работы по процессорам), возможно более высокое ускорение. Проблемы этого вида называют проблемами масштабируемости [43].

Во-вторых, объем осуществляемой работы должен быть распределен равномерно по процессорам. Это еще более необходимо, если во время вычислений некоторые данные подлежат обмену, или некоторые процессы должны быть синхронизированы. В этом случае, некоторые из процессоров должны простаивать в ожидании, пока другие не закончат. Распределение нагрузки, являющееся весьма легким для некоторых задач, остается довольно для других сложной проблемой. Объем работы на каждом процессоре мог бы варьироваться в течение вычисления, например, когда некоторый критический пункт должен быть исследован ближе. Таким образом, прекрасный баланс на одном шаге не означает баланс в следующем.

Этого можно было бы легко избежать, перемещая часть работы к другому, недостаточно загруженному процессору. Но это всегда означает коммуникацию между различными процессорами. И коммуникация, увы, отнимает много времени.

Это приводит нас к третьей проблеме. Почти во всех "реальных" задачах, процессоры регулярно нуждаются в информации друг друга для продолжения своей работы. Это имеет два эффекта. Как упомянуто выше, для этих обменов неотъемлемыми являются пункты синхронизации, заставляющие некоторые из процессоров ждать других. И конечно сама коммуникация занимает некоторое время. В зависимости от архитектуры, обмен даже одним битом информации между двумя процессорами может быть во много раз медленнее, чем доступ к нему непосредственно на процессоре. Максимальное время обмена одним байтом может занять почти столько же времени,

как и обмен сотнями байтов. Доступным или нет может быть обмен между двумя не соседствующими парами процессоров.

Так как все это очень машинно-зависимо, очевидно, что проблему (или данные) желательно разделить таким образом, чтобы не только работа распределена была равномерно, но и время коммуникаций минимизировано. Как мы увидим дальше, для определенных задач это приведет к проблеме разбиения графа на партиции. В самом простом случае это означает, что мы распределяем вершины графа на равные по размеру части таким образом, чтобы число ребер, соединяющих эти части было минимальным.

1.2 Разбиение графов

Сначала, некоторые примечания (см. [34,61]):

Граф (V, E) состоит из множества вершин $V = \{v_i \mid i = 1,\ldots,n\}$, некоторые из которых связаны ребрами множества $E = \{ e_{i,j} = (i,j) \mid$ *есть ребро между* v_i *и* $v_j\}$. Другими названиями вершин являются узлы, точки сетки или сети.

Чтобы избегать чрезмерных подстрочных индексов, мы будем иногда использовать i вместо v_i и (i,j) вместо $e_{i,j}$, если это не будет вызывать путаницы.

Подмножество $\bar{\upsilon} \subset \upsilon$ вершин, *порожденный подграф* $(\bar{\upsilon}, \bar{\varepsilon})$ содержит все те ребра $\bar{\varepsilon} \subset \varepsilon$, которые соединяют любые две вершины из множества $\bar{\upsilon}$.

И вершины, и ребра могут иметь веса, связанные с ними.

$W_e = \{w_e(e_{i,j}) \in N \mid e_{i,j} \in E\}$ - вес ребра, и $Wv = \{w_v(v_i) \in N \mid v_i \in V\}$ - вес вершины соответственно.

Веса – это неотрицательные целые числа. Благодаря этому часто возможно проводить более эффективный анализ или реализацию алгоритмов. Например, сортировка списка целых чисел ограниченного определенной величиной может быть осуществлена, используя алгоритм сортировки с вычисляемыми адресами (bucket-sort) с его производительностью порядка $O(n)$. Но это ограничение не является столь значительным, каким кажется. Например, положительные рациональные веса могут быть легко отображены в N_+, масштабируя их по наименьшему общему делителю. Если не указаны какие-либо веса, все они полагаются равными 1.

Самая простая форма разбиения графа, *невзвешенное деление графа пополам,* которое заключается в следующем: исходный граф, состоящий из вершин и ребер, необходимо разделить на два множества вершин равного размера таким образом, чтобы число ребер между этими двумя множествами было минимальным. Формально это запишется так:

Допустим, G = (V, E) - граф с количеством вершин |V| (число элементов множества V).

Необходимо найти такое разбиение (V_1,V_2) множества V (что выполняются $V_1 \cup V_2 = V$ и $V_1 \cap V_2 = 0$. Такое *непересекающееся* объединение иногда обозначается как $V_1 \dot{\cup} V_2 = V$) с равным количеством вершин

$$|V1| = V2|,$$

что число элементов из множества E

$$|\{e_{i,j} \in E \mid v_1 \in V_1 \text{ и } v_2 \in V_2\}| \quad (1.1)$$

было минимальным среди всех возможных разбиений V на части, равного размера.

В более общую постановку задачи включены взвешенные ребра, взвешенные вершины и p непересекающихся подмножеств. Тогда сумма весов вершин в каждом из подмножеств должна быть одинаковой, а суммарный вес ребер между этими подмножествами минимизирован. Опишем опять задачу более формально:

Имеется граф G = (V, E) с весами W_v и W_e (при этом, $\sum_{v_i \in V} w_v(v_i)$ делится на p частей).

Необходимо найти p-разбиение $(V_1, V_2,\ldots,V_p)$ множества V, чтобы:

$$\bigcup_{i=1}^{p} V_i = V \text{ и } V_i \cap V_j = \varnothing \text{ для всех } i \neq j$$

и

$$\sum_{v(i)\in V_j} w_v(v_i) \text{ был равным для всех } j \in \{1,2,\ldots,p\},$$

а

$$\sum_{\substack{e_{i,j}\in E \\ v_i\in V_p, v_j\in V_q, \\ p\neq q}} w_e(e_{i,j}) \quad (1.2)$$

был минимальным среди всех возможных разбиений V.

(1.1) и (1.2) также именуются *величиной разреза (cut-size)*, подразумевая под этим количество ребер исходного графа, которые не входят в подграфы, основанные на полученных подмножествах вершин, либо их суммарный вес. Другими используемыми названиями являются *сечение ребер* (*edgecut)* или *стоимость разбиения (cost of the partition).*

Как описано выше, мы ищем небольшое подмножество ребер, удалив которое, граф разделяется на два непересекающихся подграфа. Этот поиск носит название поиска ребер разделителей (*finding an edge separator*).

Вместо ребер, можно также искать подмножество вершин, так называемых вершин-разделителей, которые разделяют граф:

Задан граф G = (V, E), необходимо найти три непересекающихся подмножества V_1, V_2 и V_s, для которых выполняется:

- $V_1 \dot{\cup} V_2 \dot{\cup} V_s = V$
- $|V_s|$ ничтожно мало
- $|V_1| \approx |V_2|$
- ни одно ребро не соединяет V_1 с V_2

В зависимости от приложения, необходимы либо ребра разделители, либо вершины разделители. К счастью, преобразовать эти два вида разделителей друг в друга несложно.

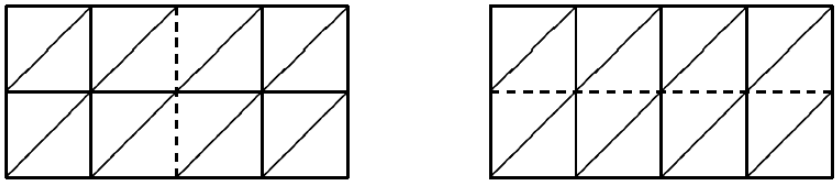
Например, предположим, что мы имеем ребра-разделители, но нуждаемся в вершинах-разделителях. Рассмотрим граф $\overline{G}$, состоящий только из отдельных ребер и их конечных вершин. Любое покрытие вершин $\overline{G}$ (то есть, любой набор вершин, который включает, по крайней мере, одну конечную вершину каждого ребра G) является множеством вершин-разделителей графа. Так как $\overline{G}$ является двудольным, мы можем вычислить наименьшее покрытие вершины эффективно двудольным соответствием.

1.3 Примеры

В следующих примерах мы предполагаем, что у нас имеется параллельный компьютер с распределенной памятью, то есть, каждый процессор имеет свою собственную память. Если процессору необходимо получить доступ к данным из памяти другого процессора, используется коммуникация. Многие параллельные компьютеры сегодня этого типа.

1.3.1 Дифференциальные уравнения в частных производных

Множество методов решения дифференциальных уравнений в частных производных (PDEs) сеточно-ориентированы, то есть данные определены в дискретной сетке узлов, конечных элементов или конечных объемов, и вычисление заключается в применении определенных операций над данными, связанными со всеми узлами, элементами или объемами сетки [61]. В этих вычислениях обычно участвуют некоторые данные соседних элементов.



good partition bad partition

Иллюстрация 1.1: Два разбиения

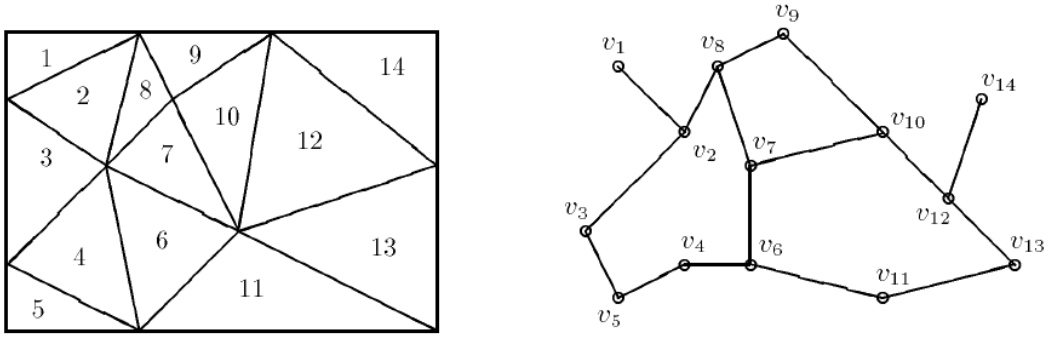


Иллюстрация 1.2: Конечная элементная сетка и соответствующий граф зависимости

Распределение проблемы на параллельных процессорах производится путем разделения сетки на подсети и решения подпроблем на различных процессорах. Для элементов на "границе" подсеток для доступа к данным на других процессорах необходима коммуникация. Используя разумное допущение, что количество коммуникаций между двумя соседями является постоянным, необходимо минимизировать "длину границы", то есть число соседей в различных разделах. (см. Иллюстрация 1.1)

Чтобы сформулировать проблему как проблему разбиения графа, посмотрим на так называемый граф зависимости сетки (см. Иллюстрация 1.2).

Каждый узел сетки (или конечный элемент) имеет соответствующую вершину на этом графе. Вес вершины пропорционален суммарной вычислительной работе, производимой на этом узле сетки. (Весьма часто веса всех вершин равны).

Для каждой пары соседних (или зависимых) узлов сетки, соответствующие вершины связаны ребром. Вес этого ребра пропорционален сумме коммуникаций между этими двумя узлами сетки. (Снова, весьма часто веса всех ребер равны).

Для конечных элементных сеток (finite element mesh) (сетка - граф, уложенный в (обычно) двух - или трехмерном пространстве так, что координаты вершин известны.), невзвешенный граф зависимости эквивалентен *двойному графу* сетки.

Теперь оптимальное разделение сетки может быть найдено, решая проблему разбиения графа на графе зависимости и распределяя узлы сетки или конечный элементы на *j-тый* процессор, если соответствующая ему вершина находится в *j-том* разделе графа.