АЛГОРИТМЫ БАЛАНСИРОВКИ ЗАГРУЗКИ ПРОЦЕССОРОВ ПАРАЛЛЕЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Бельков Д.В.
Донецкий национальный технический университет, г. Донецк
кафедра вычислительной математики и программирования

Abstract

Belkov D.V. The load balancing algorithms for parallel computer systems. *An important practical task, arising up on the stage of parallel computing, which consists in the load processors balancing, is decided in article. The algorithms for solving of this task are proposed.*

Общая постановка проблемы

Использование параллельных вычислений при решении сложных задач позволяет существенно снизить временные затраты. Одной из важных задач, возникающих при создании распределенной вычислительной среды, является оптимальная балансировка нагрузки компьютеров. Дисбаланс нагрузки может возникнуть по нескольким причинам. Алгоритмический дисбаланс вызывается особенностями алгоритмов. Универсального метода борьбы с ним не существует. Известны, например, методы балансировки в системах распределенных баз данных [1,2].

При параллельных вычислениях требуется передавать данные между узлами. Задержки в очередях к узлам могут привести к коммуникационному дисбалансу, при котором параллельная программа работает медленнее своего последовательного аналога.

Аппаратный дисбаланс возникает, если вычислительная система объединяет неоднородные аппаратные или программные ресурсы.

Методы балансировки могут быть разделены на статические и динамические. Статические методы не учитывают текущее состояние нагрузки. План размещения вычислительных заданий по узлам распределенной системы составляется до начала ее работы. Динамические методы следует применять, если время выполнения задачи намного превышает время необходимое на балансировку. В общем случае динамическая задача балансировки должна включать в себя не только распределение текущей нагрузки по процессорам, но и выбор их оптимального числа в соответствии с особенностями вычислительного алгоритма. Балансировка нагрузки может выполняться программно или аппаратно, централизованно или децентрализовано.

Постановка задачи балансировки нагрузки (многопроцессорное расписание) приведена в работе [3]. Задано множество заданий T, число процессоров m, длительности выполнения каждого задания и общий директивный срок D. Необходимо составить m–процессорное расписание для заданий из T, которое удовлетворяет общему директивному сроку D. Задача является NP-полной, если параллельная программа содержит неоднородный цикл и выполняется на неоднородной вычислительной системе.

В задачах балансировки нагрузка оптимально распределяется между процессорами согласно следующим критериям:

1. максимально равномерное распределение задач по всем процессорам;
2. максимальная производительность системы для всех приложений;
3. минимальное время ответа системы на каждый запрос;

максимально быстрое выполнение задач согласно расписанию.

Многие современные промышленные суперкомпьютеры имеют встроенные системы балансировки нагрузки, работающие на уровне приложений, например, Load leveler, Cluster System Management, DYNAMITE. Однако, как отмечается в работах [4,5], необходимость

повышения эффективности параллельных вычислений делает актуальной разработку новых алгоритмов балансировки загрузки процессоров.

Постановка задач исследования

В данной статье логическая структура задачи, которую решает параллельная система, моделируется в виде связного графа, где вершины представляют вычислительные задания, из которых состоит задача, а ребра — взаимосвязи между заданиями.

Для балансировки загрузки процессоров параллельной системы необходимо решить следующие основные задачи:

– сформировать подграфы исходного графа задачи таким образом, чтобы минимизировать число обрезанных ребер между подграфами;

– распределить подграфы среди процессоров с целью балансировки загрузки процессоров.

Решение задач и результаты исследований

Предлагаемый метод балансировки загрузки процессоров параллельной системы состоит из двух этапов. На первом этапе формируются подграфы исходного графа задачи таким образом, чтобы минимизировать число обрезанных ребер между подграфами. На втором этапе полученные подграфы распределяются среди процессоров с целью балансировки загрузки процессоров.

Рассмотрим первый этап балансировки. Обозначим: Θ — директивное время решения задачи, граф которой содержит m вершин, т.е. задача состоит из m вычислительных заданий, n — максимальное число процессоров, S — максимально допустимое число обрезанных ребер, C_j — суммарное число обрезанных ребер j–го подграфа, T_{ij} — время выполнения i–го задания на j–м процессоре, $X_j = 1$, если формируется j–й подграф, иначе $X_j = 0$, $A_{ij} = 1$, если i–я вершина графа входит в j–й подграф: $\sum_{k=1}^{i-1} T_{kj}X_j + T_{ij}X_j \le \Theta$ и $C_j \le S$, иначе $A_{ij} = 0$.

На данном этапе балансировки загрузки процессоров необходимо минимизировать число обрезанных ребер каждого подграфа. Задача минимизации обрезанных ребер изоморфна известной задаче о покрытии.

Целевая функция:

$$\sum_{j=1}^{n} C_j X_j \to \min \qquad (1)$$

Ограничения:

$$\sum_{j=1}^{n} A_{ij} X_j > 1,\ i=1,2,\ldots,m \qquad (2)$$

В задаче (1)–(2) необходимо найти матрицу A и вектор X, которые обеспечивают минимум целевой функции (1) при ограничениях (2).

Для решения задачи можно использовать жадный алгоритм A1 с временной сложностью $O(mn)$:

Шаг 1. Переменной i присвоить значение единица, элементам матрицы A и вектора X присвоить значение нуль.

Шаг 2. Переменным A_{ij} и X_j присвоить значение единица, если

$$\sum_{k=1}^{i-1} T_{kj}X_j + T_{ij}X_j \le \Theta$$

Шаг 3. Переменной i присвоить значение (i+1).

Шаг 4. Если $i \le m$, то перейти к шагу 2, иначе перейти к шагу 5.

Шаг 5. Вычислить значение целевой функции и завершить алгоритм.

Известно [6], что жадные методы решения задачи о покрытии асимптотически оптимальны.

Перейдем ко второму этапу балансировки загрузки процессоров. Обозначим: Θ — среднее директивное время решения задачи, n — число подзадач (подграфов, на которые разбит граф задачи), равное числу процессоров, T_{ij} — среднее время решения i–й подзадачи на j–м процессоре, $X_{ij} = 1$, если i–я подзадача решается на j–м процессоре, иначе $X_{ij} = 0$.

Задача оптимизации балансировки загрузки имеет вид:

Целевая функция:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} T_{ij}X_{ij} \to \max \qquad (3)$$

Ограничения:

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad (4)$$

$$\sum_{i=1}^{n} T_{ij}X_{ij} \le \Theta \qquad (5)$$

В задаче (3)–(5) максимизируется время, в течение которого процессоры не простаивают. Ограничение (4) означает, что каждая подзадача назначается на один из процессоров. Ограничение (5) означает, что суммарное время решения не превышает директивного. Максимизация целевой функции (3) при условиях (4) и (5) обеспечивает сбалансированное решение подзадач, т.к. величина $\sum_{i=1}^{m} T_{ij}X_{ij}$ ограничена значением Θ справа и максимально стремится к значению Θ слева.

Для решения задачи (3)–(5) можно использовать жадный алгоритм А2, с временной сложностью $O(n^2)$:

Шаг 1. Формируется матрица $T(n \times n)$

Шаг 2. Задаются нулевые значения матрице $X(n \times n)$ и вектору $W(n)$

Шаг 3. Полагается i=1;

Шаг 4. Пока $i \le n$:

а) Формируется вектор $Z(n)$: $Z_j = 1$, если $W_j + T_{ij} \le \Theta$, иначе $Z_j = 0$

б) Среди узлов, для которых $Z_j = 1$, находится узел p такой, что $T_{ip} = \max\limits_{Z_j=1} T_{ij}$

в) Назначается задание i в узел p: $X_{ij} := 1$, $W_p := W_p + T_{ij}$

г) Полагается i=i+1 и осуществляется переход к шагу 4а

Шаг 5. Сформированная матрица X используется для определения целевой функции по формуле (3).

При большом количестве заданий относительную погрешность данного алгоритма можно вычислить следующим образом. Если A — значение целевой функции, полученное алгоритмом А2, M — максимально возможное значение целевой функции, полученное при $\Theta = \infty$, то максимальная относительная погрешность алгоритма А2: $Q_m = (M - A) \cdot 100\% / M$.

Для исследования работы алгоритма А2 проведен вычислительный эксперимент. В первой серии экспериментов. Для каждого значения m, указанного в таблице 1 с помощью

алгоритма А2 решено 10 задач размещения m заданий по n узлам при n=10. Средняя относительная погрешность алгоритма вычислялась по формуле: $Q_c = 0.1 \cdot \sum_{k=1}^{10} Q_m^k$, где Q_m^k — максимальная относительная погрешность алгоритма при решении задачи k. Результаты экспериментов показаны в таблице 1 и на рисунке 1.

Таблица 1 — Погрешность первой серии экспериментов

№	m	Q_c, %
1	15	11,62
2	20	10,38
3	25	8,02
4	50	7,39
5	75	5,45
6	100	5,09
7	250	2,65
8	500	1,79
9	750	1,59
10	1000	1,21

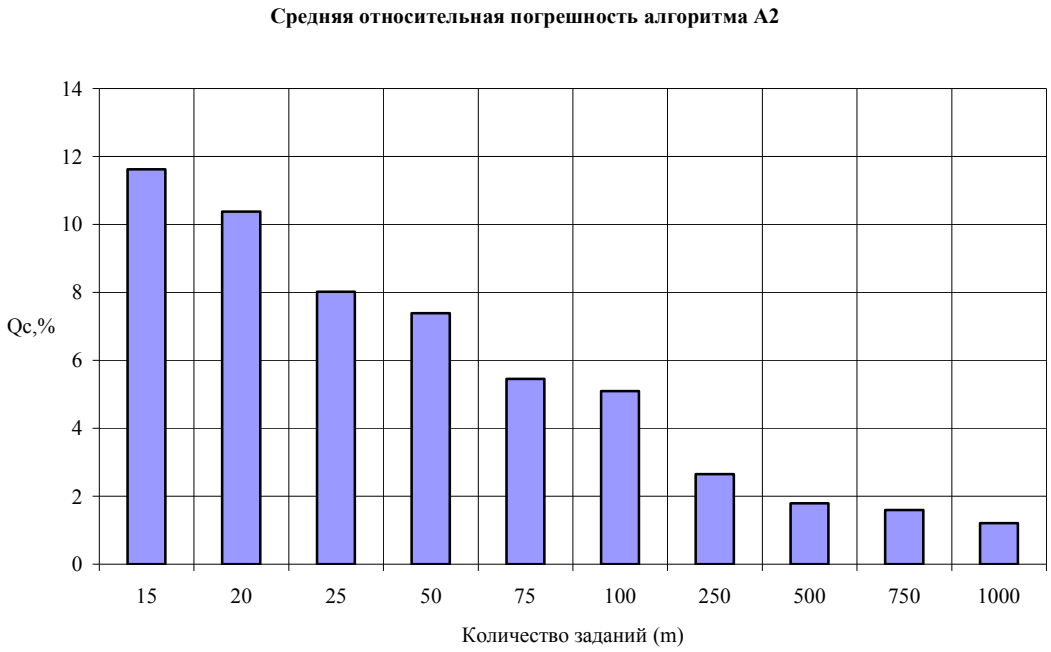

Рисунок 1 — Средняя относительная погрешность алгоритма А2

Во второй серии экспериментов с помощью алгоритма А2 решено 10 задач размещения m заданий по n узлам, $n=10^3$, $m=5 \cdot 10^3 - 5 \cdot 10^6$. Зависимость среднего времени решения от числа заданий показана на рисунке 2. При $m > 10^5$ время решения растет экспоненциально. Для реальных компьютерных сетей (число заданий меньше 10^5) трудоемкость алгоритма линейно зависит от числа заданий.

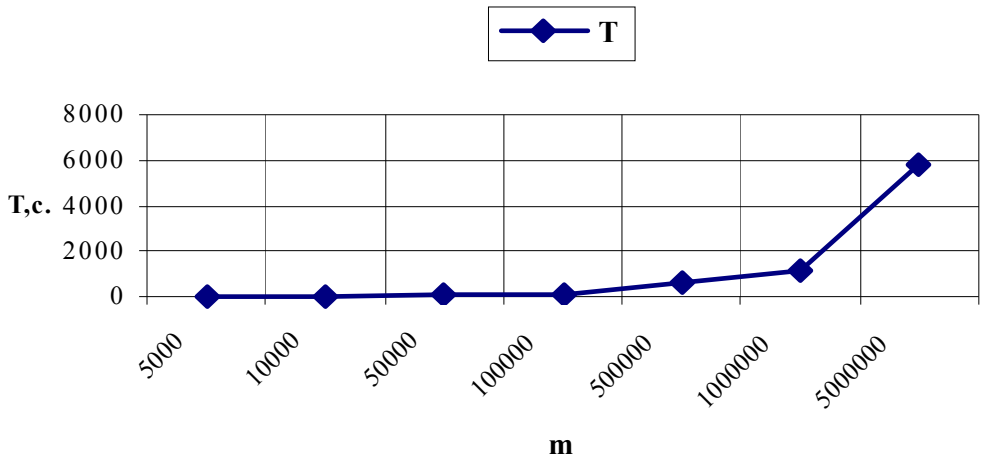

Рисунок 2 — Зависимость среднего времени решения от числа заданий

Выводы

1. Сформулированы две задачи, последовательное решение которых обеспечивает балансировку загрузки процессоров параллельной системы: задача минимизации числа обрезанных ребер между подграфами и задача максимизации времени, в течение которого процессоры не простаивают.

2. Предложены жадные алгоритмы решения сформулированных задач. При решении тестовых задач порядка 1000 заданий и 10 процессоров относительная погрешность алгоритмов не превысила 2%.

Предложенные в работе алгоритмы могут быть использованы в системах централизованной балансировки загрузки типа Master/Slave [7]. Перспективным направлением является разработка динамических методов балансировки загрузки процессоров на основе данных алгоритмов.

Литература

1. Соколинский Л.Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой. //Программирование. — 2001. — № 6. — С. 13–29.
2. Соколинский Л.Б. Обзор архитектур параллельных систем баз данных. //Программирование. — 2004. — № 6. — С. 49–63.
3. Гери М., Джонсон Д. Вычислительные машины и трудно решаемые задачи. Москва: Мир, 1982. — 367 с.
4. Бухановский А.В., Иванов С.В. Особенности оптимизации распределения вычислительной нагрузки в задачах параллельной обработки информации и имитационного моделирования. //www.gpss.ru/immod’03/013.html
5. Rotaru T., Nageli H.H. Heterogeneous dynamic load balancing with a scheme based on the Laplasian polynomial. Lecture notes in computer sciences. Reading, 2001. — P. 307.
6. Кузюрин Н.Н. Задача линейного булева программирования и некоторые комбинаторные проблемы. //Компьютер и задачи выбора. М.: Наука, 1989. — С. 144–160.
7. Hendricson B., Devine K. Dynamic load balancing in computational mechanics. //www.cs.sandia.gov/~bahendr/partitioning.