Об одном алгоритме балансировки вычислительной нагрузки в распределенных системах*

А.С. Хританков

В статье предлагается алгоритм балансировки нагрузки для распределенных систем с расписанием, в которых узлы доступны только часть времени решения задачи. При построении алгоритма использовалась модель характеристик производительности распределенных систем. Также, в статье приводится оценка эффективности предлагаемого алгоритма.

1. Введение

В данной статье представлен алгоритм распределения одинаковых по трудоемкости задач между вычислителями распределенной системы, минимизирующий полное время решения. Под распределенной системой понимается совокупность вычислителей различной производительности объединенных вычислительной сетью, которые доступны для решения задач только часть времени. *Вычислительными системами с расписанием* называются распределенные системы, для которых расписание выделения вычислителей известно после завершения решения. Предлагаемый алгоритм разработан для интервальных систем. Интервальными системами будем называть системы с расписанием, в которых вычислители объединены в кластеры. Вычислители, принадлежащие одному кластеру, работают по одному расписанию, и планирование подзадач осуществляется для всех вычислителей кластера. Например, невозможно назначить задачу одному из вычислителей кластера отдельно от других. В этом состоит главное отличие модели интервальных систем от традиционной модели коллектива вычислителей, в которой вычислители управляются независимо друг от друга. Модель интервальной системы может применяться при оценке характеристик производительности распределенных систем, включающих вычислительные кластеры, управляемые системами пакетной обработки, а также Грид-систем, в которых управление отдельными вычислителями не допускается. Предлагаемый алгоритм учитывает данную особенность в управлении вычислителями при планировании подзадач.

Проблема планирования задач, вообще говоря, является NP сложной даже для неоднородных систем. В данной работе предлагается решение задачи планирования с целью минимизации времени расчета для более широкого класса систем в предположении, что все подзадачи имеют одинаковую трудоемкость. Благодаря тому, что предлагаемый алгоритм обладает линейной по числу подзадач сложностью, он может быть использован, например, для предварительной оценки времени решения оптимизационных задач с помощью методов Монте-Карло.

Традиционная модель производительности параллельной системы как «черного ящика» описана в книге [1]. Среди моделей производительности неоднородных и распределенных систем можно упомянуть работу [2], в которой предложена модель производительности для сети разнородных рабочих станций, работающих в режиме разделения времени. В работе предложены понятия эффективности и ускорения для указанного класса систем на основе понятия работы, совершаемой системой и ее вычислителями. Интересная модель производительности неоднородных систем была предложена в книге [3].

2. Модель производительности распределенных систем

Используемая в качестве основы модель систем с расписанием была представлена в работах [4-5]. Здесь приводится краткое ее описание, используемое при дальнейшем изложении. По

* Работа выполнена при финансовой поддержке программы № 15П фундаментальных исследований Президиума РАН «Разработка фундаментальных основ создания научной распределенной информационно – вычислительной среды на основе технологий GRID» и проектов РФФИ №06-07-89079-а, №08-07-00072-а.

определению, под распределенной вычислительной системой понимается совокупность n вычислителей, объединенных вычислительной сетью и работающих во времени, при этом набор вычислителей, выделенных для решения некоторой задачи A, может изменяться во время решения. Обозначим время решения задачи A через T. Выделение i-го вычислителя для решения задачи A задается расписанием, описываемым функцией $h_i(t)$. Функция $h_i(t)$, равна 1, если вычислитель выделен для решения задачи, и 0 в противном случае. Для каждого вычислителя полагается известным эталонное время $\overline{T}_i$ решения задачи A с помощью некоторого фиксированного эталонного алгоритма. Эталонное время $\overline{T}$ решения вычислительной системой задачи A определяется как минимальное время t, удовлетворяющее соотношению:

$$\int_0^t \sum_{j=1}^{n} \frac{h_j(\tau)}{\overline{T}_j} d\tau = 1.$$

Для построенной модели определяются понятия доступности устройства ρ_i, эффективности E и относительного ускорения (speedup) устройства S_i:

$$\rho_i(T) = \frac{1}{T}\int_0^T h_i(t)dt, \quad S_i = \frac{\overline{T}_i}{T}, \quad E = \frac{\overline{T}}{T}.$$

Заметим, что параллельную систему можно рассматривать как частный случай распределенной системы, в которой все устройства одинаковы и выделены для решения задачи A на все время решения, тогда введенные характеристики совпадают с традиционными определениями. Модель может применяться для оценки производительности распределенных и GRID систем (описанных, например, в книге [6]).

3. Постановка задачи балансировки нагрузки

Пусть имеется задача A, декомпозируемая на несколько подзадач A_i, $i = 1..W$ одинаковой трудоемкости $L(A_i) = a$. При этом трудоемкость решения исходной задачи A равна сумме трудоемкостей решения составляющих ее подзадач: $L(A) = \sum_{i=1}^{W} L(A_i) = Wa$. Без ограничения общности можно положить $a \equiv 1$.

Рассмотрим вычислительную систему с расписанием, состоящую из N вычислителей. Каждый вычислитель принадлежит одному из кластеров c_k из множества кластеров C, $c_k \in C$. Количество вычислителей в кластере k обозначим через n_k, при этом будем полагать, что только c_k, $c_k < n_k$, вычислителей будут задействованы для решения подзадач. Остальные вычислители обеспечивают их работу. Полагается, что вычислители кластера k работают согласно общему для них расписанию $h_k(t)$, $k \in 1..|C|$, и имеют одинаковую эталонную производительность $\pi_k = \frac{a}{\overline{T}_j}$, $\overline{T}_j = \tau_k$, $j \in c_k$, при решении подзадач A_i.

Задачи A_i назначаются для решения на кластеры пакетами по несколько задач. Каждый вычислитель может решать одновременно не более одной задачи, поэтому на кластер может быть назначено для решения в данный момент времени не более c_k задач. Кластер начинает решать задачи из следующего пакета только после окончания решения уже назначенных задач. Промежуток времени решения кластером пакета задач будем называть *этапом решения кластера*. Для каждого вычислителя кластера промежуток времени этапа решения кластера будем называть *этапом решения вычислителя*. Без ограничения общности можно полагать, что каждый кластер работает только один интервал $[t_k^0, t_k^1]$, а последующие интервалы отождествлять с работой модельных кластеров такой же конфигурации.

Вычислительные системы с расписанием такого рода будем называть *интервальными вычислительными системами.* Время начала интервала работы кластера k обозначим через t_k^0, время окончания интервала через t_k^1. Максимальное количество этапов решения J_k на вычислителе кластера k и суммарное число задач R, которое могут решить все вычислители, составляют

$$J_k = \left\lfloor \frac{t_k^1 - t_k^0}{\tau_k} \right\rfloor,\ R = \sum_{k=1}^{N} c_k J_k . \qquad (1)$$

Время окончания k-го этапа ($j = 1..J_k$) на кластере k выражается формулой

$$t_{kj} = t_k^0 + j\tau_k . \qquad (2)$$

Упорядочим этапы по возрастанию значения t_{kj} и перенумеруем, сопоставив этапу i величину t_i, $t_i \ge t_j$, $i > j$, $i = 1..R$, $j = 1..R$. Определим вектор $\vec{s} \in \{0,1\}^R$ так, что $s_i = 1$, если какая-либо подзадача назначена для решения на этапе i. Время решения задачи A по определению равно наибольшему времени завершения решения одной из ее подзадач A_k вычислителями системы. Пусть подзадача A_k решена на этапе i со временем окончания t_i. Задача нахождения распределения подзадач по этапам решения вычислителей для минимизации времени решения исходной задачи имеет вид:

$$\min_{\vec{s}} \max_{i} s_i t_i , \qquad (3)$$

$$\sum_{i=1}^{R} s_i \le W ,\ \vec{s} \in \{0,1\}^R .$$

Ограничение неравенством указывает на то, что число решенных задач должно быть не больше числа имеющихся задач. Решением сформулированной задачи, как будет показано далее, является вектор $\vec{s}^* = (1,..1,0,..0)$, в котором $s_i^* = 1,\ i = 1..W$.

Сформулируем задачу максимизации суммарной "производительности" вычислительной системы с учетом расписания.

$$\max_{\vec{s}} \sum_{i=1}^{R} \frac{s_i}{t_i}, \qquad (4)$$

$$\sum_{i=1}^{R} s_i \le W ,\ \vec{s} \in \{0,1\}^R .$$

Решением приведенной задачи, очевидно, является вектор $\vec{s}^* = (1,..1,0,..0)$, в котором $s_i^* = 1$, $i = 1..W$, максимизирующий сумму

$$\sum_{i=1}^{R} \frac{s_i}{t_i}.$$

Утверждение 1. Пусть заданы времена окончания этапов вычислений $t_i > 0, i = 1..R$, $t_i \ge t_j, i > j$ .. Тогда задача (3) эквивалентна задаче (4).

Обозначим через S множество всех этапов, для которых $s_k = 1$. В дальнейшем изложении величину $\pi_i = 1/t_i$ будем называть приведенной производительностью этапа i. Вектор s^* является решением задачи (4) тогда и только тогда, когда сумма производительностей π_i используемых этапов для произвольного вектора s, не больше суммы производительностей, используемых этапов для вектора s^*. Вектор s^* является решением задачи (3) тогда и только тогда, когда среди задействованных этапов произвольно выбранного вектора s найдется этап, производительность которого будет не больше производительностей всех этапов решения s^*. (при этом в s^* найдется, по крайней мере, один этап большей производительности, чем в s).

Покажем, что решение задачи (3) также является решением задачи (4). Из условия задачи следует, что производительности всех этапов для s^* не меньше производительностей всех этапов для вектора s. Действительно, допустим противное, что найдется такой вектор s', для которого производительность одного из этапов k' больше производительности какого-либо этапа k^* из s^*. Тогда мы можем составить новый вектор s'', совпадающий с вектором s^* во всех чего?, кроме одного этапа, замененного на этап из s' большей производительности. Так как общее число компонент вектора осталось прежним, он также является решением задачи. При этом для вектора s^* найдется этап k'' производительности не большей, чем все этапы для вектора s'.

Покажем, что каждое решение задачи (4) будет также решением задачи (3). Вектор $s^* \in \{0,1\}^R$ является решением задачи (4), тогда и только тогда, когда сумма производительностей этапов из соответствующего ему множества S^* не меньше суммы производительностей этапов для произвольного вектора $s \in \{0,1\}^R$:

$$\sum_{i \in S^*} \pi_i s_i^* \geq \sum_{i \in S} \pi_i s_i$$

Допустим, что найдется такой вектор $s' \neq s^*$, что для него выполняется указанное соотношение и среди этапов множества S^* имеется этап k^* производительности меньшей, чем все этапы из S'. Выберем этап k' из множества $S' \setminus S^*$. Тогда можно построить новый вектор s'' соответствующий множеству $S'' = S^* \setminus \{k^*\} \cup \{k'\}$, при этом суммарная производительность этапов из множества S^* будет меньше суммы производительностей этапов из множества S'', откуда следует, что вектор s^* не является решением. Мы пришли к противоречию, поэтому предположение о существовании вектора s' неверно. Следовательно, среди этапов всех векторов s найдется этап производительности, не превышающей производительности этапов решения s^*, то есть вектор s^* также является решением задачи (3).

4. Алгоритм балансировки нагрузки

Алгоритм *вычисления распределения задач по этапам* основан на решении задачи (4), согласно утверждению 1, эквивалентной исходной задаче (3).

Алгоритм 1. Алгоритм вычисления распределения задач по этапам:

1. По формулам (1) рассчитывается количество этапов J_k для каждого кластера и суммарное число этапов R.
2. Для каждого этапа по формуле (2) рассчитывается время окончания t_{kj}.
3. Для минимизации времени решения задачи A необходимо выбрать W этапов с наименьшим временем окончания и распределить подзадачи для решения на этапах.

Сложность алгоритма составляет $O(R)$, если на шаге 3 выбор осуществлять при помощи метода, основанного на алгоритме быстрой сортировки [7]. Суть алгоритма состоит в том, чтобы использовать алгоритм быстрой сортировки для нахождения W-го по номеру элемента, применяя алгоритм только к той области набора, границы которой охватывают элемент с номером W. При использовании алгоритма быстрой сортировки со случайным выбором разделяющего элемента, размер областей, сортируемых алгоритмом при рекурсии, уменьшается экспоненциально в среднем, поэтому построенный алгоритм выбора W элементов в среднем потребует только $O(R)$ операций.

Если обозначить количество задач, назначенных для решения на этапе j кластера k через f_{kj}, $k = 1..|C|$, $j = 1..J_k$, $0 \leq f_{kj} \leq c_k$, то решение задачи (3) с помощью приведенного алго-

ритма позволяет получить значения f_{kj} для всех этапов решения, а также ожидаемое время решения задачи T_e. На основе полученного распределения задач по этапам f_{kj} построим *алгоритм балансировки* и управления вычислительным пространством без обратной связи.

Алгоритм 2. Алгоритм балансировки:

1. Рассчитать f_{kj} с помощью алгоритма вычисления распределения задач по этапам.
2. Для каждого кластера k вычислить назначенное число задач $s_k = \sum_{j=1}^{J_k} f_{kj}$.
3. В процессе решения задач, если кластер k простаивает, передать на кластер $\min\{c_k, s_k\}$ задач и уменьшить s_k на число переданных задач.
4. Если для кластера k больше задач не осталось, то есть $s_k = 0$, остановить кластер k.

Оценим величину *структурной неэффективности* системы при распределении задач согласно предложенному алгоритму. Структурной неэффективностью называется падение эффективности системы вследствие невозможности управления отдельными ее элементами. При этом другие причины падения эффективности учитывать не будем. Например, вследствие наличия вспомогательных вычислителей в кластере при $c_k < n_k$, эффективность системы будет снижена, так как вспомогательные вычислители не решают задачи. Другим примером структурной неэффективности является невозможность назначить задачи всем вычислителям кластера, если недостаточно доступных задач.

Заметим, что алгоритм обладает двумя важными свойствами:

1. Все этапы решения кластера, кроме последнего, полностью заполнены задачами.
2. Время решения всех задач на данном расписании выделения кластеров минимально.

Воспользуемся первым свойством для оценки эффективности алгоритма. Вследствие сделанных предположений, падение эффективности присутствует только на тех этапах, на которых часть вычислителей кластера простаивает. При использовании данного алгоритма вычислители могут простаивать только на последнем этапе. Пусть последний этап решения соответствует кластеру s, обозначим номер этого этапа через j_s.

Обозначим эффективность, достигаемую системой, через E^*. Снижение эффективности составляет $\Delta E = 1 - E^*$. По определению, эффективность рассчитывается по формуле $E^* = \overline{T}^* / T^*$. В качестве оценки времени решения задачи воспользуемся ожидаемым временем завершения $T^* = T_e$:

$$T_e = t_{sj_s} = t_s^0 + j_s \tau_s.$$

Эталонное время решения оценим сверху. Эталонное время складывается из времени начала интервала выделения t_s^0 кластера s, суммы времен всех этапов решения, кроме последнего, $(j_s - 1)\tau_s$ и времени эталонного решения задач последнего этапа вычислителями кластера. При этом не учитывается, что часть вычислительной нагрузки может быть передана на другие кластеры.

$$\overline{T}^* \le t_s^0 + (j_s - 1)\tau_s + \frac{f_{sj_s}}{n_s}\tau_s.$$

После подстановки и группировки членов, выражение для оценки структурной неэффективности будет иметь вид:

$$\Delta E \ge \left(1 - \frac{f_{sj_s}}{n_s}\right)\frac{\tau_s}{T_e}.$$

С другой стороны, эталонное время не может быть меньше, чем время начала последнего этапа решения, так время начала этапа является эталонным временем для меньшего количества задач, а эталонное время возрастает с увеличением числа задач при неизменном расписании.

$$\Delta E \le \frac{\tau_s}{T_e}.$$

Итак, структурная неэффективность системы лежит в пределах:

$$\left(1 - \frac{f_{sj_s}}{n_s}\right)\frac{\tau_s}{T_e} \le \Delta E \le \frac{\tau_s}{T_e}.$$

4. Заключение

В статье представлены результаты исследований по управлению вычислительным пространством в распределенных и Грид-системах, полученные к настоящему моменту. В работах [4, 5, 9] была представлена модель производительности, в рамках которой был разработан предлагаемый алгоритм. Применимость модели интервальных систем для оценки характеристик производительности распределенных систем была проверена на примере распределенной системы BNB-Grid [8]. В статье [9] приводятся результаты экспериментов и оценки эффективности и ускорения при решении задачи конформации молекул [10].

Необходимо исследовать, насколько существенным ограничением предложенного алгоритма является требование одинаковой трудоемкости задач, распределяемых между вычислителями системы. Влияние данного ограничения планируется исследовать экспериментально и теоретически. Одним из направлений исследований является изучение модели интервальных систем для задач, трудоемкость которых может быть описана случайной величиной с одинаковым распределением.

Литература

1. A. Grama, A. Gupta, G. Karypis, V. Kumar. Introduction to Parallel Computing, Second Edition. – USA: Addison Wesley, 2003
2. Zhang, X., Yan, Y. Modeling and characterizing parallel computing performance on heterogeneous networks of workstations. // In Proceedings of the 7th IEEE Symposium on Parallel and Distributed Processing (October 25 - 28, 1995). SPDP
3. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – С.П.:БХВ-Петербург. 2002
4. Хританков А.С. Математическая модель характеристик производительности распределенных вычислительных систем // Избранные труды 50-й научной конференции МФТИ. – 2007.
5. Хританков А.С. Оценка характеристик производительности распределенных вычислительных систем // Труды XV Научной конференции студентов, аспирантов и молодых ученых «Ломоносов» – 2008. – с. 53-54.
6. Проблемы вычислений в распределенной среде: организация вычислений в глобальных сетях. Труды ИСА РАН / под. ред. Емельянова С.В., Афанасьева А.П. – М.: РОХОС, 2004.
7. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. - М.:МЦНМО. – 2004.
8. А.П. Афанасьев, В.В. Волошинов, М.А. Посыпкин, И.Х. Сигал, Д.А. Хуторной "Программный комплекс для решения задач оптимизации методом ветвей и границ на распределенных вычислительных системах" // Труды ИСА РАН, 2006, Т. 25, С. 5-17.
9. Посыпкин М.А., Хританков А.С. О понятии ускорения и эффективности в распределенных системах // Труды Всероссийской научной конференции Научный сервис в сети Интернет: решение больших задач. – 2008. – с.149-155.
10. D. J. Wales, J. P. K. Doye "Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms" // Journal of Physical Chemistry, №101, 1997, С. 5111-5116.