Модели и алгоритмы распределения нагрузки

Модель коллектива вычислителей. Модели с соперником

А.С. Хританков

Аннотация. В работе представлен обзор класса моделей коллектива вычислителей, предназначенных для оценки характеристик производительности вычислительных систем. Модели данного класса позволяют рассчитать часто используемые характеристики производительности, например, эффективность и коэффициент ускорения. Предложен новый класс моделей с соперником. Дан краткий обзор диффузионных моделей и алгоритмов распределения нагрузки.

Ключевые слова: балансировка вычислительной нагрузки, распределенные и параллельные вычисления, модели с соперником, модель коллектива вычислителей, диффузионные алгоритмы и модели.

Введение

Данная работа является первой частью обзора различных направлений в моделировании распределенных систем и алгоритмов балансировки вычислительной нагрузки.

В разделе 1 представлен ряд моделей для оценки характеристик производительности вычислительных систем от наиболее простой модели системы как «черного ящика» до более сложных моделей, описывающих структуру системы. Данные модели позволяют определить, насколько эффективно вычислительная система решает какую-либо одну задачу. Примером модели данного рода может служить модель коллектива вычислителей.

Диффузионные алгоритмы и алгоритмы парного обмена, представленные в разделе 3, являются классическими в сфере балансировки нагрузки в больших параллельных системах, однако могут быть использованы и при распределении нагрузки в системах массовых вычислений и Грид-системах.

Достаточно новым направлением в балансировке нагрузки являются модели производительности с соперником. Подход к описанию системы, в котором алгоритму балансировки «противостоит» соперник, был разработан для моделирования наихудших ситуаций при маршрутизации в коммуникационных сетях, так как характеристики производительности, рассчитанные с помощью теории массового обслуживания, достаточно сильно зависели от закона распределения длительности интервалов времени между последовательными заявками во входящем потоке и распределений времен обработки заявок маршрутизаторами. При использовании алгоритмов, обладающих согласно теории массового обслуживания хорошей производительностью, система иногда оказывалась в перегруженном состоянии. Модели с соперником позволяют исследовать поведение системы в данном случае.

Игровые методы в моделировании вычислительных систем используются для анализа пиринговых и мультиагентных систем. Ключевым понятием метода является понятие равновесия в системе, достигаемого в результате взаимодействия агентов. Взаимодействие независимых агентов приводит к некооперативной игре

и возможности использования равновесия Нэша для индивидуальных агентов [6]. Описанию парадоксов, возникающих в транспортных сетях и сетях массового обслуживания при использовании некоторых алгоритмов балансировки при помощи игровых методов, посвящена статья [5]. В работе [4] рассказано о применении экономических моделей к анализу распределенных систем.

1. Модель коллектива вычислителей

1.2. Модель однородной вычислительной системы без структуры

Широко распространенная модель параллельных вычислительных систем описана в [7]. Вычислительная система рассматривается как совокупность вычислительных элементов, способных решать некоторую общую задачу. Структура связей между узлами не рассматривается. Описываемая модель чаще всего применяется к однородным параллельным системам, узлами которых обычно являются процессоры, поэтому в данном разделе для краткости будем называть вычислительные элементы системы процессорами.

В рамках модели вводится несколько характеристик производительности. Временем T_p параллельного выполнения алгоритма (решения задачи) называется длина промежутка времени между началом и завершением решения задачи системой. Временем накладных расходов T_o называется суммарное время, затраченное узлами на организацию и реализацию всех обменов данными между ветвями параллельного алгоритма. Временем последовательного решения T_s называется время решения задачи на параллельной системе из одного процессора. Пусть в системе имеется n процессоров, тогда время накладных расходов выражается формулой:

$$T_o = nT_p - T_s .$$

Коэффициентом накладных расходов называется

$$\varepsilon = \frac{T_o}{T_p} .$$

Коэффициент ускорения параллельного алгоритма вычисляется по формуле

$$S = \frac{T_s}{T_n} .$$

Закон Амдала выражает связь между долей последовательных вычислений δ в параллельном алгоритме и максимально достижимым коэффициентом ускорения $S*$:

$$S \le S*,$$

$$S* = \frac{1}{\delta + (1-\delta)/n},$$

где n - число узлов в системе.

Коэффициентом эффективности однородной вычислительной системы называется величина

$$E = \frac{S}{n} .$$

Как правило, ускорение, из-за наличия накладных расходов меньше числа узлов $S \le S* \le n$, поэтому коэффициент эффективности меньше единицы $E \le 1$.

«Парадокс» параллелизма состоит в возможности сверхлинейного ускорения и достижении коэффициента эффективности более единицы при решении задачи на параллельной системе: $S > n,\ E > 1$. Данный эффект обусловлен несколькими факторами, основные из которых:

- эффект памяти, когда задача не помещается в памяти процессора при последовательном решении и в результате обращений к диску время решения увеличивается;

- эффект параллельного поиска, когда, например, при поиске в графе один из процессоров находит искомый элемент; при этом полное число операций может быть меньше, чем при последовательном выполнении поиска, вследствие иного порядка просмотра элементов графа.

Рассмотрим важную характеристику параллельной системы - изоэффективность [7]. Отметим два обстоятельства влияющих на эффективность параллельных систем:

1. для заданного размера задачи при увеличении числа процессоров эффективность системы падает;

2. во многих случаях эффективность параллельной системы увеличивается с ростом размера задачи при неизменном числе процессоров.

На основании приведенных соображений, вычислительную систему будем называть мас-

штабируемой, если ее эффективность остается постоянной при увеличении числа процессоров и размера задачи. Под размером задачи W понимается количество элементарных операций, необходимых для решения задачи лучшим последовательным алгоритмом на каком-либо процессоре системы.

Время параллельного решения задачи может быть выражено через функцию накладных расходов $T_o(W,n)$:

$$T_p = \frac{W + T_o(W,n)}{n}.$$

Выражения для коэффициента ускорения и эффективности принимают вид:

$$S = \frac{Wn}{W + T_o(W,n)},$$

$$E = \frac{1}{1 + T_o(W,n)/W}.$$

Если увеличивать число процессоров при постоянном размере задачи, то из последнего выражения видно, что эффективность падает, так как растут накладные расходы $T_o(W,n)$. Если же увеличивать размер задачи при постоянном числе процессоров, то эффективность масштабируемых систем растет, так как для них функция накладных расходов $T_o(W,n) = O(W)$ при постоянном n.

Пусть в системе требуется поддерживать некоторую эффективность E'. Обозначим $K = E'/(1-E')$, тогда из выражения для эффективности следует, что $W = KT_o(W,n)$.

Если данное соотношение удается разрешить относительно W, то можно получить явное выражение для размера задачи как функции числа процессоров, при котором эффективность системы будет равна заданному значению E':

$$W = I(K,n).$$

Функция $I(K,n)$ называется функцией изоэффективности системы.

В качестве примера рассмотрим алгоритм параллельного сложения l чисел на системе из n процессоров. Алгоритм вначале производит сложение примерно l/p чисел на каждом из процессоров, а затем полученные суммы складывает между собой. Процесс сложения сумм логически организован в виде бинарного дерева. Время последовательного выполнения составляет $T_s = \Theta(l)$, параллельного $T_p = \Theta(l/n + 2\log n)$, поэтому коэффициент ускорения составляет

$$S = \Theta\left(\frac{nl}{l + 2n\log n}\right).$$

Функция накладных расходов на организацию взаимодействия процессоров $T_o(W,n)$ определяется выражением по порядку величины

$$T_o = nT_p - T_s = l + 2n\log n - l = 2n\log n.$$

Функция изоэффективности для данного алгоритма составляет

$$I(W,n) = 2Kn\log n.$$

Следовательно, увеличивая размер задачи согласно функции изоэффективности, в данной системе мы сохраним заданное значение эффективности. Такая система является масштабируемой.

1.2. Модель коллектива вычислителей

Модель коллектива вычислителей позволяет описать структуру связей между элементами системы. Под коллективом вычислителей будем понимать совокупность программно-аппаратных вычислителей, объединенных в вычислительную систему. Модель коллектива вычислителей [9] представляет собой совокупность описания структуры вычислительной системы H и алгоритма работы коллектива вычислителей A:

$$S = < H, A >.$$

Структура коллектива вычислителей H описывается в виде:

$$H = < C, G >,$$

где $C = \{c_i\}$ - множество вычислителей c_i, $i = 1..N$, G - описание структуры связей между вычислителями. Модель коллектива вычислителей отражает следующие основные принципы:

- параллельное выполнение операций вычислителями из множества C и их взаимодействие через связи, описываемые с помощью G;

- программируемая структура вычислительной системы, вычислители являются универсальными, решаемые задачи определяются программным обеспечением;

- однородность конструкции H, включающая однородность коллектива вычислителей C и однородность структуры G.

Структура коллектива вычислителей представляется графом $G =< C, E >$. Вершинам графа сопоставлены вычислители из C, ребрам графа E соответствуют связи между вычислителями.

При организации высокопроизводительных вычислительных систем, состоящих из большого числа вычислителей-процессоров, скорость обмена данными между вычислителями может стать узким местом системы. Специальные виды организации вычислителей позволяют увеличить пропускную способность и уменьшить задержки при обмене информацией. Выделяют две важные характеристики графа, соответствующего структуре связей между вычислителями: порядок вершины неориентированного графа равен количеству ребер, инцидентных ему; диаметром графа называется максимальное число вершин на пути между двумя произвольными вершинами графа. Структуры, соответствующие графам с меньшим диаметром, обеспечивают меньшие задержки при передаче данных, так как меньшее число вычислителей участвуют в передаче данных между источником и получателем. Наименьшим диаметром обладает полный граф. Однако при достаточно большом числе вычислителей в системе (современные параллельные системы включают 10^0–10^6 вычислителей) реализовать полный граф при помощи аппаратных средств невозможно вследствие большого числа связей каждого вычислителя с другими, равного числу вычислителей в системе. Возникает задача минимизации диаметра графа, описывающего структуру связей, при сохранении небольшого порядка вершин. Оптимальными в этом смысле графами являются графы Мура (Moore) [10]. Частными случаями графа Мура являются, например, граф Петерсена и граф Хоффмана-Синглтона. Примеры использования алгоритмов глобальной оптимизации для поиска оптимальной конфигурации связей вычислителей в параллельных системах можно найти в работах [11,12]. При разработке вычислительных систем используются следующие виды структур связей между вычислителями:

- нульмерные структуры, в которых взаимодействие между вычислителями осуществляется посредством общей шины (например, кластеры рабочих станций, объединенных с помощью Ethernet);

- решетки, одномерные – линейные, двумерные и большей размерности;

- тороидные структуры различной размерности (структура связей BlueGene/L является трехмерным тором);

- гиперкубические структуры, обладающие достаточно простой структурой и достаточно небольшим диаметром (порядка логарифма от числа вычислителей) при большом количестве вычислителей в системе;

- иерархические структуры, например деревья, используются при построении распределенных и GRID-систем.

Алгоритм A работы коллектива вычислителей S обеспечивает согласованную работу вычислителей и связей между ними в процессе решения общей задачи. Алгоритм может быть формально представлен в виде

$$A = A(P(D)),$$

где D - исходные данные для параллельной программы P. В модели коллектива вычислителей обычно полагается, что каждый вычислитель выполняет одну и ту же программу, но разные ее ветви в зависимости от исходных данных. Поэтому массив исходных данных представим в виде объединения индивидуальных массивов данных D_i для каждого вычислителя

$$D = \bigcup_{i=1}^{N} D_i .$$

Параллельная программа представляется как совокупность ветвей P_i, выполняемых разными вычислителями:

$$P = \bigcup_{i=1}^{N} P_i, \ \bigcap_{i=1}^{N} P_i = \varnothing .$$

Данный подход к организации параллельных вычислений называется *параллелизм по задачам* (MIMD, согласно классификации Флинна) – вычислители выполняют разные программы на различных входных данных.

Модель коллектива вычислителей позволяет сформулировать некоторые принципы построения вычислительных систем.

Однородность – принцип построения систем, согласно которому все вычислители обладают одинаковой функциональностью и способны решать общий круг задач. Однородность системы может быть достигнута при реализации принципа модульности системы, согласно которому вычислительная система строится из унифицированных элементов, взаимодействующих через строго определенные интерфейсы.

Близкодействие или локальность – принцип построения вычислительных систем, при котором вычислители могут взаимодействовать только с ограниченной частью других вычислителей. В структуре связей вычислителей, описываемой графом G, локальность обеспечивается указанием связей между вычислителями при помощи ребер графа. Взаимодействие с остальными вычислителями происходит с помощью промежуточных вычислителей, расположенных в вершинах графа на пути, соединяющем исходный и конечный вычислители.

Принцип близкодействия допускает реализацию механизма управления вычислительной системой, который не зависит от количества составляющих ее вычислителей. При этом поведение каждого вычислителя зависит от поведения только ограниченного подмножества других вычислителей.

1.3. Модель системы функциональных устройств

Модель коллектива вычислителей была обобщена на неоднородные вычислительные системы в виде модели системы функциональных устройств [13] и распределенных систем с расписанием [14].

Модель коллектива вычислителей может быть использована для анализа параллельных однородных вычислительных систем, однако неприменима непосредственно к неоднородным вычислительным системам, узлы-вычислители которых могут различаться по характеристикам. Для преодоления ограничения однородности необходимо ввести индивидуальные характеристики вычислителей.

Примером построения модели неоднородных вычислительных систем может служить модель системы функциональных устройств, предложенная в [13]. Под вычислительной системой **R** понимается совокупность n работающих во времени функциональных устройств, способных решать задачу A. Автор рассматривает вычислительную задачу как набор элементарных вычислительных операций, а ее решение - как выполнение элементарных операций, составляющих задачу. Полагается, что все срабатывания одного устройства являются одинаковыми по длительности, то есть выполняются устройством за одинаковое время. Каждая операция обладает стоимостью, под которой подразумевается время выполнения операции. Стоимостью работы называется сумма стоимостей всех выполненных операций, то есть время последовательного выполнения данных операций на устройствах с аналогичными временами срабатывания.

Далее вводится понятие *реальной производительности* системы **R** как количества элементарных операций, выполняемых устройствами системы в среднем в единицу времени. По-видимому, под этим подразумевается следующее. Пусть время решения задачи A равно T и задача A состоит из N элементарных операций, тогда реальная производительность системы равна

$$r = \frac{N}{T}.$$

Под *пиковой производительностью* понимается максимальное количество операций, которое может быть выполнено той же системой за единицу времени при отсутствии связей между устройствами. Данное определение, по-видимому, следует понимать как максимальное суммарное число операций, которое выполняют устройства системы, если каждое из них решает задачу A независимо от других. Пусть пиковая производительность i-го устройства равна π_i, тогда, по определению, пиковой производительностью системы будет называться величина

$$\pi = \sum_{i=1}^{n} \pi_i .$$

С другой стороны, по определению пиковой производительности системы ее величина равна

максимальному числу операций в единицу времени. Возможно, подразумевается, что существует некоторая дифференцируемая функция стоимости $V(t): R^+ \to R^+$, значение которой равно числу элементарных операций, выполненных к данному моменту времени, и максимальное значение ее производной является значением пиковой производительности

$$\pi = \max_t \frac{dV(t)}{dt}.$$

Либо следует несколько раз решить задачу *A* на системе **R** и выбрать максимальное значение реальной производительности в качестве пиковой

$$\pi = \max_k \{r_k\}.$$

В книге не приводится метод определения пиковой производительности. *Загруженностью* устройства на отрезке времени называется отношение стоимости выполненной за это время работы к максимально возможной стоимости. Здесь, вероятно, идет речь о функции стоимости $V(t)$ и под загруженностью понимается

$$p_i(t_1, t_2) = \frac{\pi \cdot (t_2 - t_1)}{V(t_2) - V(t_1)}.$$

Заметим, что данное соотношение не приводится в [13] и является лишь возможной интерпретацией.

Загруженностью системы называется взвешенная сумма

$$P = \sum_{i=1}^{n} \alpha_i p_i, \ \alpha_i = \frac{\pi_i}{\pi}.$$

Для однородных параллельных систем коэффициент ускорения определяется как отношение времени решения задачи одним устройством ко времени решения задачи вычислительной системой [7,9]:

$$S = \frac{T_0}{T}.$$

Для неоднородных систем вследствие того, что устройства различны, коэффициент ускорения в [13] определяется по отношению к устройству с наибольшей пиковой производительностью:

$$S = \frac{\sum_{i=1}^{n} p_i \pi_i}{\max_i \pi_i}.$$

Другим подходом к определению коэффициента ускорения является введение векторного коэффициента [14]. Коэффициент эффективности неоднородной системы полагается равным загруженности системы.

1. 4. Модель системы с расписанием

Модель системы функциональных устройств была обобщена на случай, когда устройства могут быть доступны для решения задачи только часть времени в соответствии с расписанием [14]. Модель системы с расписанием позволяет описывать распределенные вычислительные системы, использующие системы пакетной обработки для разграничения доступа к вычислительным ресурсам.

В модели системы с *расписанием* для каждого устройства задано расписание – функция времени $h_i(t)$ такая, что $h_i(t) = 1$, если устройство в момент времени *t* выделено для решения, и $h_i(t) = 0$ в противном случае. *Доступностью* $\rho_i(t)$ устройства называется доля интервала времени $[0, t]$, в течение которого устройство было выделено для решения задачи:

$$\rho_i(T) = \frac{1}{T} \int_0^T h_i(t) d\tau.$$

В отличие от модели системы функциональных устройств, понятие пиковой производительности не используется, вместо этого вводится понятие *эталонного времени решения* $\overline{T}_i > 0$ *задачи A i-м устройством*, полученное с использованием эталонного алгоритма, и *эталонной производительности* π_i устройства *i* при решении задачи *A*, по определению равное $\pi_i = L / \overline{T}_i$, где *L* – трудоемкость задачи. Трудоемкость задачи выражает наше априорное знание о сложности решения задачи, т.е. о вычислительных ресурсах, которые необходимо затратить на ее решение. Трудоемкость задачи может рассматриваться как число элементарных операций, затрачиваемых лучшим последовательным алгоритмом на решение задачи [7].

Вычислительной системой с расписанием **R** называется совокупность

$$\mathbf{R} = < \vec{\pi}, \vec{h}(t) >, \vec{\pi} = (\pi_1, .., \pi_n), \vec{h}(t) = \left(h_1(t), \ldots, h_n(t)\right).$$

Эталонная производительность системы с расписанием определяется как сумма эталонных производительностей устройств, выделенных для решения задачи в данный момент времени *t*:

$$\pi(t) = \sum_{i=1}^{n} \pi_i h_i(t).$$

При полной доступности устройств $h_i(t) \equiv 1$ в процессе решения задачи и при одинаковой эталонной производительности узлов $\pi_i = \pi_0$, эталонная производительность системы будет совпадать с $n \cdot \pi_0$, то есть с эталонной производительностью однородной параллельной системы, определяемой в традиционной модели производительности.

Эталонным временем решения $\overline{T}$ задачи системой **R** называется время ее решения системой при условии, что все узлы работают с эталонной производительностью, задача может быть разделена между узлами произвольным образом и определяется следующим соотношением:

$$\overline{T} = \arg\min_{t} \left\{ \int_{\tau=0}^{t} \pi(\tau) d\tau = L \right\}.$$

Тогда естественно определить эффективность *E* системы **R** как отношение эталонного и реального времен решения задачи:

$$E = \frac{\overline{T}}{T},$$

где *T* – реальное время решения задачи *A*.

Ускорение для параллельной системы определяется как отношение времени решения задачи на одном устройстве ко времени решения задачи на всей системе. В системе с расписанием устройства могут быть различными, поэтому вводится понятие относительного ускорения. *Ускорением S системы* $\mathbf{R}_1$ *относительно системы* $\mathbf{R}_2$ называется отношение времен решения задачи этими системами: $S(\mathbf{R}_1, \mathbf{R}_2) = T_2 / T_1$. По аналогии, ускорением S_i называется отношение эталонного времени решения задачи на устройстве *i* ко времени решения задачи *A* на всей системе: $S_i = \overline{T}_i / T$. Под относительным ускорением системы с расписанием понимается вектор $\vec{S} = (S_1, S_2, .., S_n)$.

Для распределенной системы с расписанием **R** справедливо соотношение:

$$E = \left(\sum_{i=1}^{n} \frac{\overline{\rho}_i}{S_i} \right)^{-1}, \text{ где } \overline{\rho}_i = \rho_i(\overline{T}).$$

2. Модели и алгоритмы распределения нагрузки, использующие соперника (Adversarial Models)

2.1. Модель коллектива вычислителей в дискретном времени

Рассматривается проблема динамического распределения задач между узлами вычислительной системы. В отличие от статической постановки проблемы балансировки, когда в системе задано только начальное распределение задач между узлами системы, в динамической системе задачи могут добавляться в процессе работы системы. Под состоянием системы будем подразумевать расположение задач по узлам. Для анализа работы системы и исследования характеристик алгоритмов балансировки нагрузки будем использовать простую модель вычислительной системы. Вычислительную систему будем описывать при помощи графа, в котором вершинам соответствуют узлы системы, а ребрам – каналы передачи данных. Эта модель известна как модель коллектива вычислителей [23].

В данном разделе предлагается обзор моделей вычислительных систем, в которых добавление задач в систему выполняется соперником (adversary). Алгоритм балансировки нагрузки, работающий в системе, стремится устранить неоднородности в загруженности узлов системы, возникающие в результате действий соперника. Теория моделей с соперником начала развиваться в начале 90-х годов прошлого века в связи с исследованиями проблем маршрутизации в динамических системах. К исследованию алгоритмов балансировки модели систем с соперником стали применяться в конце 90-х для изучения устойчивости локальных алгоритмов балансировки нагрузки в распределенных системах [15]. В настоящее время модели с соперником применяются для исследования устойчи-

вости алгоритмов балансировки при различных алгоритмах работы соперников.

В системе имеется некоторое количество задач, представляемых абстрактными метками. Процесс работы системы рассматривается по шагам, которые называют раундами. На каждом раунде метка находится на одном из узлов, что моделирует решение узлом соответствующей метке задачи. Загруженность узла в данном раунде определяется как число меток, расположенных на нем. Распределение меток происходит путем передачи меток через ребра соседним узлам. Цель распределения нагрузки состоит в поддержании системной устойчивости, то есть такого распределения меток между узлами, чтобы отклонения количества меток на узлах от среднего были ограничены.

Особый интерес представляют локальные алгоритмы распределения нагрузки. В отличие от централизованного подхода к управлению перемещением меток, здесь каждый узел сравнивает собственную загруженность с загруженностью своих соседей и принимает решение о передаче меток на основе этой информации. Задачей алгоритма балансировки является перераспределение меток между узлами для достижения баланса за минимальное число раундов. Если при решении задач в системе они добавляются в процессе работы, то для балансировки нагрузки используются динамические алгоритмы. Динамический алгоритм балансировки называется устойчивым, если максимальное отклонение загруженности узлов от средней загруженности по системе ограничено равномерно во времени. Одним из подходов к исследованию динамических алгоритмов является применение методов теории вероятностей и случайных процессов, когда подразумевается существование некоторого случайного процесса, добавляющего и извлекающего задачи из системы (например, [21,22]).

Описанная модель является синхронной в том смысле, что раунды на всех узлах происходят в одно и то же время; локальной, так как узлы могут взаимодействовать только с соседними узлами; распределенной в том смысле, что алгоритмы управления системой, например, балансировки нагрузки, выполняются на всех узлах.

В литературе можно выделить два направления исследований алгоритмов балансировки нагрузки с помощью моделей с соперником. В первом случае соперник может только добавлять задачи [17-19]. Подобный способ более всего схож с моделями, используемыми при анализе алгоритмов маршрутизации. В рамках другого направления соперник может как добавлять задачи, так и убирать их [15,16]. При этом не делается каких-либо предположений о размере задач, их сложности и производительности узлов.

2.2. Модели с соперником для задач неизвестной сложности

Рассмотрим модель, описанную в [15]. Соперник в начале каждого раунда может добавлять метки на узлы, что соответствует новым задачам, а также убирать метки с узлов, что соответствует окончанию решения задач. После этого алгоритм балансировки может перемещать метки через ребра графа, не более одной за раунд, для достижения сбалансированного распределения нагрузки. Чередование действий продолжается бесконечное число раундов. Заметим, что, позволяя сопернику не только добавлять, но и убирать задачи, мы моделируем наихудший вариант, когда задачи имеют неизвестное время выполнения до тех пор, пока задача не будет решена.

Силой соперника будем называть число $r \geq 0$ такое, что соперник для произвольного подмножества $S \subseteq V$ может изменить число меток в нем не более чем на $r \cdot e(S)$, где $e(S)$ - величина разреза между S и $\neg S$. При помощи теоремы о максимальном потоке можно показать, что при $r > 1$ любой алгоритм балансировки неустойчив [15].

Обозначим через a_t среднее число меток в системе по узлам в начале раунда *t,* а через $h_t(v),\ v \in V$ - загруженность или высоту узла, вычисляемую как число меток на нем. Алгоритм балансировки называется стабильным, если отклонение $b_t(v) = |a_t - h_t(v)|$ равномерно ограничено сверху некоторой константой $B>0$. Далее рассмотрим простой инициируемый отправителем алгоритм, вопросы сбора данных

о состоянии соседних узлов не будем принимать во внимание.

На действия соперника вводится ограничение:

$$|\delta_t(S) - |S|(a_{t+1} - a_t)| \le r\alpha |S|, r < 1,\ S \subseteq V,$$

где $\delta_t(S) = \sum_{v \in S}(h_{t+1}(v) - h_t(v))$ - изменение числа меток, α - величина *расширения графа по ребрам* (edge expansion), равная минимальному по всем подмножествам $S \subseteq V$ числу исходящих из него ребер (величине разреза).

Пусть *d* - максимальное число ребер, инцидентных узлу в графе. Рассматривается следующий локальный алгоритм:

Алгоритм SimpleLocal:
для каждого t, для каждого ребра $e = (u, v)$
Если $h_t(u) - h_t(v) \ge 2d + 1$, то передать метку от *u* к *v*
Если $h_t(v) - h_t(u) \ge 2d + 1$, то передать метку от *v* к *u*

При этом действия алгоритма в остальных случаях не влияют на полученные в статье результаты. Для данного алгоритма доказана следующая теорема [15].

Теорема 1. Для произвольного $\varepsilon > 0$ алгоритм SimpleLocal устойчив при сопернике силы $r = 1 - \varepsilon$.

Данный результат был улучшен в работе [16]. Задается некоторый пороговый параметр $\theta > 0$ и рассматривается более общий локальный алгоритм:

Алгоритм $SCLB_\theta$:
для каждого t, для каждого ребра $e = (u, v)$
Если $h_t(u) - h_t(v) \ge \theta$, то передать метку от u к v
Если $h_t(v) - h_t(u) \ge \theta$, то передать метку от v к u

На действия соперника накладывается ограничение, называемое ограничением разреза (cut condition):

$$|\delta_t(S) - |S|(a_{t+1} - a_t)| \le |e(S)|,\ S \subseteq V.$$

Данное ограничение соответствует $r = 1$. Справедливо следующее утверждение.

Теорема 2. Для произвольно определенного соперника, соблюдающего ограничение разреза (силы не более 1), и любого $\theta \ge 1$ алгоритм $SCLB_\theta$ устойчив, то есть существует константа $B > 0$, зависящая от начального числа меток θ и G такая, что $|b_t(v)| \le B,\ v \in V,\ t > 0$.

Полученный результат был обобщен: а) на случай систем с динамической структурой, для которых множество ребер *E* может изменяться со временем, при этом на соперника накладывается ограничение:

$$|\delta_t(S) - |S|(a_{t+1} - a_t)| \le |e_t(S)|,\ S \subseteq V;$$

б) на случай, когда соперник может подчиняться ограничению разреза в рамках некоторого периода времени – окна (window) размера *W*, определяемого для произвольного момента времени *t* как интервал $[t, t+W)$, условие, накладываемое на соперника, будет иметь вид:

$$|\sum_{r=t}^{t+W}\delta_r(S) - |S|(a_{t+W} - a_t)| \le W \cdot |e(S)|,\ S \subseteq V.$$

В наиболее общей постановке, исследованной в [16], при целой пропускной способности ребер, динамическом графе системы и при подчинении в рамках произвольного фиксированного количества последовательных раундов, условие на действия соперника имеет вид:

$$|\sum_{r=t}^{t+W}\delta_r(S) - |S|(a_{t+W} - a_t)| \le W \cdot |e_t(S)|,\ S \subseteq V,$$

где $e_t(S)$ - величина разреза между *S* и $\neg S$ на раунде *t*. При указанных условиях будет справедлива теорема 2.

2.3. Модели с предположениями о сложности задач

В [18] исследованы свойства вычислительной системы в стационарном состоянии (подразумевается, что оно существует) при использовании простого стохастического локального алгоритма балансировки. Система состоит из *n* узлов, соединенных сетью произвольной топологии. Процесс работы системы рассматривается как последовательность раундов из двух фаз. В первой фазе каждого раунда задачи размещаются на узлах детерминированным либо стохастическим соперником, обладающим полной информацией о текущем и всех прошлых состояниях системы. В отличие от [15,16], соперник может только добавлять задачи в систему, но не убирать их. Узлы могут обмениваться за-

дачами во второй фазе раунда, при этом пропускная способность ребер не ограничивается. Задачи полагаются одинаковыми и каждый узел решает одну задачу за раунд в конце второй фазы, если в его очереди имеются задачи. Алгоритм балансировки стремится уравнять число задач на всех узлах путем обмена задачами между соседними узлами.

Рассматриваются два вида соперников: детерминированный соперник, добавляющий в систему не более λn, $\lambda < 1$, задач в каждом раунде; стохастический соперник, который на каждом раунде размещает на узлах системы n генераторов произвольным образом. Каждый генератор с вероятностью $\lambda < 1$ создает задачу и помещает ее в очередь узла, на котором он в данный момент находится.

Пусть на каждом узле имеется очередь подзадач, более старые задачи (находящиеся в системе большее время) находятся ближе к началу очереди и выполняются первыми. В работе предложен алгоритм обмена задачами с четными номерами так, чтобы число задач на обменивающихся узлах сравнялось. При этом получаемые задачи размещаются в очереди на данном узле в соответствии с их временем пребывания в системе по отношению к уже имеющимся на узле задачам. Алгоритм извлекает необходимое число задач с четными номерами из начала своей очереди для отправки партнеру по балансировке, принятые от партнера задачи он размещает между задачами в своей очереди так, чтобы задача, более близкая к началу очереди, имела большее время пребывания в системе. Схема выбора партнера по балансировке состоит в следующем. Для каждого узла *i* инцидентное ему ребро (*i, j*) добавляется в список возможных соединений для балансировки с вероятностью $1/(8\max\{d_i, d_j\})$, где d_i - количество соседних узлов узла *i*. Если ребро (*j, i*) было также выбрано узлом *j*, то узел *j* добавляется в список партнеров для балансировки узла *i* и наоборот.

Пусть задан граф системы $G =< V, E >$, максимальный порядок узла (вершины графа) равен *d*. Матрицу смежности графа *G* обозначим через *A*. Пусть $D = \{d_{ij}\}$, $d_{ii} = d_i$, $d_{ij} = 0 (i \neq j)$ - диагональная матрица порядков узлов. Напомним, что лапласианом графа *G* называется вырожденная матрица $L = D - A$. Собственные значения *L* обозначим $0 = \Lambda_1 \leq \Lambda_2 \leq .. \leq \Lambda_n$. При этом $\Lambda_2 = \Omega(n^{-2})$, если *G* - связный граф. Обозначим $\gamma = \Lambda_2 / 16d$.

Система называется *устойчивой*, если общее число задач в системе равномерно по времени ограничено некоторой константой.

Теорема 3. Вычислительная система, использующая описанный выше алгоритм, устойчива по отношению к детерминированному сопернику, при этом среднее число задач в системе составляет $O(\gamma^{-1} n \ln n)$ при $t \to \infty$. Если система стартует без нагрузки, то соотношение выполняется в каждый момент времени.

Далее в работе исследуется эффективность работы алгоритма балансировки. Под эффективностью алгоритма понимается совокупность характеристик, связанных со временем пребывания задач в системе.

Теорема 4. Обозначим через $W(t)$ случайную величину, равную среднему времени пребывания задач, поступивших на раунде *t*. Для любого c>1 найдется число $\kappa = \kappa(c)$ такое, что:

1. $\lim_{t\to\infty} P\{W(t) \leq \kappa\gamma^{-1} \ln n\} \geq 1 - n^{-c}$,
2. $\lim_{t\to\infty} E[W(t)] = O(\gamma^{-1} \ln n)$.

Указанные соотношения справедливы без знаков предельного перехода, если система стартует без нагрузки.

В [17,18] рассматривается алгоритм случайного заимствования задач (Random Stealing). Пусть в системе присутствует стохастический соперник, описанный ранее. Работа алгоритма инициируется узлом при опустошении очереди задач и состоит в передаче половины имеющихся у случайно выбранного узла задач. Таким образом, число задач у двух узлов становится одинаковым. Все задачи полагаются одинаковыми, пропускная способность ребер не ограничена.

С помощью теории марковских процессов в [17] показано, что если граф системы полный, то есть каждый узел может заимствовать задачи у любого другого узла системы, то полученная система будет устойчивой. При этом количест-

во задач в системе при $t \to \infty$ в среднем ограничено полиномиально по размеру системы. Формулировка теорем, представленных в работе, достаточно объемна, поэтому мы их здесь не приводим.

В [18] показано, что для систем с разреженной структурой и небольшим числом связей между узлами алгоритм случайного заимствования задач, исследованный в работе [17], будет неустойчив и число задач в системе будет расти неограниченно во времени либо будет экспоненциально зависеть от числа узлов в системе. Рассмотрим следующий пример. Пусть имеется система, состоящая из $n > 3$ узлов, связанных в цепочку. Пронумеруем узлы по порядку расположения. Детерминированный соперник размещает на первом узле $n-2$ задачи, на втором одну задачу на каждом раунде. В модели, представленной в [17], узлы производят обмен задачами после того, как соперник добавил задачи в очереди на узлах, но перед их решением, поэтому очередь на втором узле никогда не будет пустовать на фазе обмена задачами. Значит, первый узел не будет участвовать в обмене задачами. Количество задач на первом узле будет неограниченно увеличиваться на $n-3$ задачи за раунд. Дело в том, что простаивающие большую часть времени узлы с номерами больше второго не имеют возможности запросить задачи у первого узла. Алгоритм, предложенный в [18], инициирует передачу задач на каждом раунде независимо от числа задач в очереди, что позволяет передавать большее число задач через второй узел.

3. Диффузионные модели и алгоритмы распределения нагрузки

3.1. Диффузионные алгоритмы

Рассмотрим систему из n узлов в модели функциональных устройств. Система моделируется совокупностью функциональных устройств, попарно соединенных коммуникационными каналами. Для каждого функционального устройства задается пиковая производительность $\pi_i > 0$ и вычислительная нагрузка $w_i \geq 0$. Загруженностью устройства называется величина w_i / π_i. Пропускную способность коммуникационного канала между устройствами обозначим через $c_{ij} \geq 0$, причем $c_{ij} = 0$, когда устройства не связаны. Сопоставим системе взвешенный граф $G = (V, E)$, устройства соответствуют вершинам графа, каналы соответствуют ребрам. Ширина канала задает вес ребра, пиковая производительность устройства задает вес вершины.

Рассмотрим статическую задачу балансировки. Изначально вычислительная нагрузка распределена согласно вектору w_i^0. Суммарная вычислительная нагрузка $W = \sum_i w_i^0 = const$ в процессе распределения нагрузки. Используя локальную информацию о загруженности, необходимо перераспределить вычислительную нагрузку так, чтобы каждое устройство было загружено пропорционально его пиковой производительности:

$$w_i^* = \frac{\sum_V w_i}{\sum_V \pi_i} \pi_i .$$

Суть диффузионного алгоритма состоит в распределении нагрузки от более загруженных узлов к менее загруженным с использованием механизма передачи от одного ко многим (multiport communication). Существуют реализации диффузионного алгоритма с применением парных обменов, однако при этом алгоритм использует устаревшую информацию о загруженности соседних узлов, передавая им задачи несколько шагов подряд без обновления данных о загруженности узлов.

Под схемой метода подразумевается методика расчета объема нагрузки для передачи соседним узлам. Мы подробно рассмотрим диффузионную схему первого порядка, так как она дает достаточно полное представление о методе. Подробное описание различных схем диффузионного метода можно найти в [29]. Диффузионная схема первого порядка (FOS) была предложена в работе [30]. Рассмотрим обобщение данной схемы на случай неоднородной системы.

Пусть $B \in R^{n \times m}$, $b_{ij} \in \{0,1\}$ – матрица инцидентности графа G. Взвешенная матрица ин-

цидентности $A = BF^{-1}$, $F_{ii} = \{f_i\}$, $1/f_i$ - пропускная способность ребра *i*. Лапласианом графа *G* называется матрица $L = AA^T$. Лапласиан можно также определить через взвешенную матрицу смежности $C = \{c_{ij}\}$, если задать степень вершины $d_j = \sum_{j:\{i,j\}\in E} c_{ij}$, $D = \{D_{ii} = d_i\}$. Тогда $L = D - C = AA^T$. Пусть диагональная матрица $S = \{S_{ii} = \pi_i\}$ содержит значения коэффициентов производительности устройств. Произведение LS^{-1} называют обобщенным (взвешенным по вершинам) лапласианом графа. Итерационная схема записывается в виде:

$$w_i^{k+1} = w_i^k - \alpha \sum_{j:\{i,j\}\in E} c_{ij}\left(\frac{w_i}{\pi_i} - \frac{w_j}{\pi_j}\right),$$

здесь $\alpha \in [0,1]$ - параметр алгоритма, ограничение указывает, что по каналу нельзя передать отрицательный объем нагрузки либо больше, чем пропускная способность канала. В матричных обозначениях итерационный процесс можно переписать следующим образом:

$$w^{k+1} = (I - \alpha(D-C)S^{-1})w^k = (I - \alpha LS^{-1})w^k,$$
$$L = D - C = AA^T, \quad M = I - \alpha LS^{-1},$$
$$w^{k+1} = Mw^k.$$

Полученная матрица *M* называется *матрицей диффузии*. Суммарная нагрузка сохраняется потому, что *M* – симметрична. Для собственных значений *M* справедливо

$$Mx = (I - \alpha LS^{-1})x = x - \alpha(LS^{-1}x) =$$
$$= x - \alpha\lambda_0 x = (1 - \alpha\lambda_0)x,$$
$$\gamma_i = 1 - \alpha\lambda_i,$$

где *x* - собственный вектор LS^{-1}. Заметим, что матрица обобщенного лапласиана LS^{-1} вырождена, значит, *M* имеет собственное число $\gamma_0 = 1$. Итерационный процесс сходится, если все остальные собственные значения *M* по модулю меньше единицы, причем сходится к собственному вектору числа $\gamma_0 = 1$. Покажем, что вектор w^*, описывающий распределение нагрузки пропорционально производительности узлов, собственный для γ_0.

$$(M - I)w^* = -\alpha(LS^{-1})w^* = -\alpha LS^{-1}\frac{\sum_V w_i}{\sum_V \pi_i}\pi =$$
$$= -\alpha\frac{\sum_V w_i}{\sum_V \pi_i}(D - C)\begin{pmatrix}1\\ ..\\ 1\end{pmatrix} = 0.$$

Так как матрица *M* симметричная и вещественная, все ее собственные значения вещественны и собственные вектора образуют базис $\{\xi_i\}$ в R^n. Разложим w^k в нем.

$$w^0 = \sum_i \beta_i\xi_i = \beta_0 w^* + \sum_{i=1}^n \beta_i\xi_i,$$
$$w^k = M^k w^0 = \beta_0 w^* + \sum_{i=1}^n \beta_i(\gamma_i)^k\xi_i.$$

Достаточным условием сходимости итерационного процесса является $\max_{i=1..n} |\gamma_i| < 1 - \varepsilon$, $\varepsilon > 0$. Собственные значения λ_i лапласиана $LS^{-1} = AA^TS^{-1}$ неотрицательны, так как AA^T симметричная положительно полуопределенная матрица, а собственные значения S^{-1} положительны. Упорядочим различные значения λ_i по возрастанию: $0 = \lambda_0 < \lambda_1 < .. < \lambda_m$. Обозначим

$$\gamma = \max_i |\gamma_i| = \max\{|1 - \alpha\lambda_1|, |1 - \alpha\lambda_m|\} < 1.$$

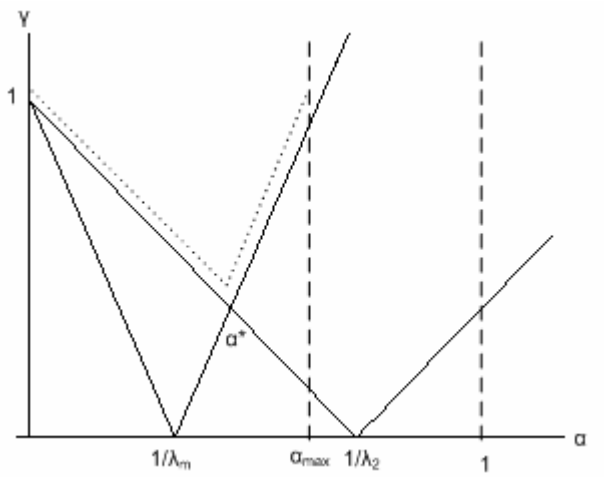
Оптимальное значение α, минимизирующее γ, равно

$$\alpha^* = \frac{2}{\lambda_m + \lambda_2}, \quad \gamma^* = \frac{\lambda_m - \lambda_2}{\lambda_m + \lambda_2}.$$

С учетом ограничений на α оптимальным выбором, как видно из рисунка, представленного выше, будет

$\alpha = \alpha^*, \gamma = \gamma^* \quad (\lambda_m + \lambda_2 \geq 2)$ — *быстрая сеть*

$\alpha = 1, \quad \gamma = 1 - \frac{1 - \gamma^*}{\alpha^*} \quad (\lambda_m + \lambda_2 < 2)$ — *медленная сеть*



К нахождению оптимального значения параметра α

Согласно теореме Гершгорина [35] об оценке собственных значений, любое собственное значение λ_k лежит в одной из областей:

$$|\lambda_k - \frac{d_i}{\pi_i}| \le \sum_{j,j\ne i} \frac{c_{ij}}{\pi_i} = \frac{d_i}{\pi_i},$$

$$0 \le \lambda_k \le 2\max_i \frac{d_i}{\pi_i} \;\Rightarrow\; \lambda_m \le 2\max_i \frac{d_i}{\pi_i} = \lambda^*.$$

Достаточным условием медленной сети является неравенство $\lambda^* \le 1$. Полученные выражения для медленной сети можно трактовать следующим образом: если отношение суммарной пропускной способности каналов на узле к его пиковой производительности для всех узлов меньше половины, то сеть нужно полагать медленной и каналы при передаче нагрузки оптимально загрузить полностью $\alpha = 1$.

Оценим скорость сходимости итерационного процесса схемы первого порядка. Начальное отклонение составляет $e^0 = w^0 - w^*$. На *k*-ом шаге отклонение от сбалансированного распределения составит $e^k = w^k - w^* = M^k(w^0 - w^*)$. В [33] показано, что для евклидовой нормы вектора ошибки e^k справедлива оценка

$$\left\|e^k\right\|_E \le \sqrt{\frac{\pi_{\max}}{\pi_{\min}}}\gamma^k \left\|e^0\right\|_E.$$

Последнее выражение получается из значения произведения спектральных норм матриц $S^{-1/2}$ и $S^{1/2}$.

Помимо схем первого порядка, использующих информацию только о предыдущем шаге итерации, существуют схемы второго и более высоких порядков. Схемы второго порядка имеет вид

$w^1 = Mw^0$,

$w^k = \beta M w^{k-1} + (1-\beta)w^{k-2},\ k = 2,3,\ldots,$

здесь β - фиксированный параметр. Оптимальное значение параметра β определяется по формуле

$$\beta = \beta_{opt} = \frac{2}{1+\sqrt{1-\gamma^2}}.$$

В данном случае скорость сходимости составит

$$\left\|e^k\right\|_E \le \sqrt{\frac{\pi_{\max}}{\pi_{\min}}}(\beta_{opt}-1)^{\frac{k}{2}}\left(1+k\sqrt{1-\gamma^2}\right)\left\|e^0\right\|_E.$$

В случае, если γ близко к единице, схема второго порядка требует $N_2 \approx \sqrt{N_1}$ шагов, где N_1 - число шагов схемы первого порядка.

В [33] показано, что рассмотренная схема распределения нагрузки перемещает минимальный объем нагрузки для достижения фиксированной ошибки. Переданная по сети к *k*-му шагу нагрузка x^k вычисляется по формуле $w^k = w^0 + AC^{-1}x^k$, где A - матрица инцидентности. Доказывается, что при использовании схем первого и второго порядков евклидова норма перемещенной нагрузки $\left\|x^k\right\|_E$ минимальна из всех возможных вариантов перемещения нагрузки по сети для конечного распределения нагрузки w^k и начального w^0. Авторы указывают, что минимальная величина перемещенной нагрузки достигается только при использовании двухэтапной схемы, когда во время первого этапа производится расчет перемещаемой нагрузки x^k с помощью итерационной схемы первого или второго порядков, а на втором этапе происходит перемещение вычислительной нагрузки.

3.2. Алгоритм парных обменов (Dimension Exchange)

Рассматривается модель коллектива вычислителей распределенной системы в дискретном времени. Все узлы системы полагаются одинаковыми, граф системы связный. Лежащая в основе алгоритма парных обменов (DE) идея состоит в балансировке каналов обмена между узлами так, чтобы не возникало перегрузок каналов и использовалась большая часть пропускной способности сети. Обмен задачами происходит между узлами в парах вдоль коммуникационных каналов, которым соответствуют ребра графа системы. Каждый узел за один раунд может обмениваться задачами только с одним другим узлом (single-port communication).

Статическая задача балансировки состоит в нахождении последовательности реберных раскрасок графа (edge-coloring) одним цветом и указания количества передаваемых задач вдоль каждого окрашенного ребра. Характеристиками производительности служат *эффективность алгоритма балансировки* как среднее по начальному распределению задач число шагов до достижения равномерного распределения задач между узлами и величина *конечной неоднородности*, равная отклонению полученного распределения задач от равномерного. Вычислительная нагрузка может рассматриваться как совокупность неделимых задач, тогда равномерное распределение достигается с некоторой допустимой погрешностью, выражаемой суммарным отклонением количества задач на каждом узле от среднего и как бесконечно делимая, тогда алгоритм стремится достичь полностью равномерного распределения нагрузки.

Алгоритм парных обменов является распределенным локальным алгоритмом, существуют различные его варианты в зависимости от того, в какой момент происходит операция балансировки, каким образом определяется число передаваемых задач и способа раскраски графа.

Алгоритм, основанный на описанной идее, был представлен в [34], согласно алгоритму узлы обмениваются половиной произвольно делимой нагрузки так, что после обмена объем вычислительной нагрузки на узлах становится одинаков. В [31] рассматривается статическая задача распределения произвольно делимой нагрузки в системе из n узлов. Подразумевается, что граф системы имеет реберную k-раскраску (раскраска в k цветов). Пусть $E = \bigcup_{c=1}^{k} E_c$, где E_c задает подмножество ребер раскраски в k-ый цвет. Нагрузка системы в момент времени t описывается вектором $l(t) = (l_1(t),..l_n(t))$.

Алгоритм GDE [31] задается kn^2 неотрицательными числами $M_{c,i,j}$, $0 \le c < k$, указывающими величину нагрузки, передаваемой от узла i узлу j при цвете c. В момент времени t параллельно происходит обмен задачами согласно раскраске E_c, $c = t \bmod k$, нагрузка на узлы i и j после операции балансировки задается формулами:

$$l_i(t+1) = M_{c,i,i} l_i(t) + M_{c,i,j} l_j(t),$$
$$l_j(t+1) = M_{c,j,j} l_j(t) + M_{c,j,i} l_i(t).$$

Данные соотношения сохраняют полную вычислительную нагрузку в системе $L = L(t) = \sum_{i=1}^{n} l_i(t)$ и для ребер, не принадлежащих E_c, обмена нагрузкой не происходит $l_i(t+1) = l_i(t)$.

Сформулируем алгоритм в терминах линейной алгебры. Пусть M_c - матрица $n \times n$, элементами которой являются $M_{c,i,j}$. Матрица раскраски M определяется как произведение $M = \prod_{c=k-1}^{0} M_c$, поэтому изменение распределения нагрузки в системе происходит согласно уравнению $l(t) = M_c l(t),\ c = t \bmod k$. Для произвольного неотрицательного q:

$l(qk) = M^q l(0)$.

Оптимальным распределением нагрузки будет равномерное распределение. Оно задается вектором $l_{ave} = (L/n,..,L/n)$. Мерой *неоднородности загруженности* выступает евклидова норма отклонения вектора распределения загруженности от равномерной:

$$\Delta(t) = \| l(t) - l_{ave} \|_E,$$

$\Delta(qk) = \left\|(M^q - U)l(0)\right\|_E$, $U = \{u_{ii}\}$, $u_{ii} = 1/n$.

В работе доказана теорема об ограничении величины неоднородности в системе после q раундов обмена, если существует k-раскраска графа системы.

Теорема 5. Существует такая константа b, зависящая только от M, что

$$\left\|M^q - U\right\|_E \le b \cdot nq^n \cdot \alpha^q,$$

где α - второе по величине собственное значение матрицы раскраски M.

В работе доказано, что в хеммингговых графах $H(g,p)$, $0 \le p \le g$, симметричный алгоритм парного обмена (SGDE, $\alpha = 1/2$) в статической постановке задачи балансировки позволяет перевести систему в сбалансированное состояние.

Теорема 6. Алгоритм SGDE переводит систему в сбалансированное состояние за k шагов на графе $H(g,p)$, при этом

$$k = \sum_{i=1}^{p} \binom{g}{i}.$$

Заметим, что в алгоритме SGDE узлы обмениваются половиной имеющейся у них нагрузки, поэтому его иногда называют простым алгоритмом парных обменов или симметричным алгоритмом.

Литература

1. Lauwereins, R., Peperstraete, J. Queuing theoretical analysis of processor utilization in parallel computers. // Comput. Syst. Sci. Eng. V. 8, N. 1, pp. 13-23, 1993
2. F. Bacelli, G. Balbo, R.J. Boncherie, J. Campos, G. Chiola. Annotated bibliography on stochastic Petri nets. // In O.J. Boxma, G.M. Koole, eds., Performance Evaluation of Parallel and Distributed Systems – Solution Methods, CWI, Amsterdam, 1994
3. T. Murata. Petri nets: Properties, analysis and applications. / Proceedings of the IEEE, V. 77, pp. 541-580, 1989
4. D. F. Ferguson, C. Nikolaou, J. Sairamesh, Y. Yemini. Economic models for allocating resources in computer systems. // In Market-Based Control: A Paradigm For Distributed Resource Allocation, S. H. Clearwater, Ed. World Scientific Publishing Co., River Edge, NJ, pp. 156-183.
5. S. Penmatsa, A.T. Chronopoulos. Cooperative load balancing for a network of heterogeneous computers. / In Proceedings of the 20th IEEE International Parallel and Distributed Symposium, p. 136, 2006
6. D. Grosu, A.T. Chronopoulos. Noncooperative load balancing in distributed systems. / J. Parallel Distrib. Comput. V. 65, N. 9, pp. 1022-1034, 2005
7. Grama, A. Gupta, G. Karypis, V. Kumar. Introduction to Parallel Computing, Second Edition. - Addison Wesley, 2003
8. N. G. Shivaratri, P. Krueger, and M. Singhal. Load Distributing for Locally Distributed Systems, IEEE Computer, December 1992, pp 33-44
9. Хорошевский В.Е. Архитектура вычислительных систем: Учеб. пособие для вузов. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2005
10. Hoffman, Alan J. & Singleton, Robert R. (1960), Moore graphs with diameter 2 and 3. // IBM Journal of Research and Development 5 (4), pp. 497–504, 1960.
11. Maekawa, M. 1981. Optimal processor interconnection topologies. In Proceedings of the 8th Annual Symposium on Computer Architecture (Minneapolis, Minnesota, United States, May 12 - 14, 1981). International Symposium on Computer Architecture. IEEE Computer Society Press, Los Alamitos, CA, 171-185.
12. G. Kotsis, "Interconnection Topologies for Parallel Processing Systems," Proc. of Parallele Systeme und Algorithmen, 1993.
13. Воеводин В.В., Воеводин Вл. В. Параллельные вычисления - С.П.: БХВ-Петербург, 2002
14. 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.
15. S. Muthukrishnan, R. Rajamaran. An Adversarial Model for Distributed Dynamic Load Balancing. / In Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures, June 1998, pp. 47-54.
16. E. Anshelevich, D. Kempe, J. Kleinberg. Stability of Load Balancing Algorithms in Dynamic Adversarial Systems. / In Proceedings of the Thirty-Fourth Annual ACM Symposium on theory of Computing, 2002, pp. 399-406
17. Berenbrink, P., Friedetzky, T., and Goldberg, L. A. The Natural Work-Stealing Algorithm is Stable. Technical Report. / UMI Order Number: CS-RR-381., University of Warwick, 2001
18. Anagnostopoulos, A. Kirsch, E. Upfal. Stability and Efficiency of a Random Local Load Balancing Protocol. / In Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, 2003, p. 472
19. Anagnostopoulos, A. Kirsch, E. Upfal. Load Balancing In Arbitrary Network Topologies With Stochastic Adversarial. / SIAM J. of Comput., V. 34, N. 3, pp. 616-639, 2005
20. W. Aiello, B. Awerbuch, B. Maggs, S. Rao. Approximate Load Balancing in Dynamic and Asynchronous Networks. / In Proceedings of the Twenty-Fifth Annual ACM Symposium on theory of Computing, 1993, pp. 632-641
21. D. Eager, E. Lazowska, J. Zahorjan. Adaptive Load Sharing in Homogeneous Distributed Systems. / IEEE Transactions on Software Engineering, 1996, V. 12.

22. M. Mitzenmacher. On the Analysis of Randomized Load Balancing Schemes. / In Proc. Of ACM Symp. on Parallel Algorithms and Architectures, 1998
23. Хорошевский В.Е. Архитектура вычислительных систем: Учеб. пособие для вузов. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2005
24. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajamaran, A. W. Richa, R. E. Tarjan, and D. Zuckerman. Tight analyses of two load balancing algorithms. / In Proc. of the 27th Annual ACM Symposium on Theory of Computing, 1995
25. Chengzhong Xu, B. Monien, R. Luling, F.C.M. Lau. An analytical comparison of nearest neighbor algorithms for load balancing in parallel computers. / In Proc. of The 9th International Parallel Processing Symposium , p. 472, 1995.
26. Corradi, A., Leonardi, L., and Zambonelli, F. Diffusive Load-Balancing Policies for Dynamic Applications. / IEEE Concurrency, V. 7, N. 1, pp. 22-31, 1999
27. R. Elsasser, B. Monien, S. Schamberger. Distributing Unit Size Workload Packages in Heterogeneous Networks. / Journal of Graph Algorithms and Applications, V. 10, N. 1, pp. 51–68, 2006
28. Hui, C. and Chanson, S. T. Theoretical analysis of the heterogeneous dynamic load-balancing problem using a hydrodynamic approach. / J. Parallel Distrib. Comput. V. 43, N. 2 , pp. 139-146, 1997
29. Elsässer, R., Monien, B., Preis, R. Diffusive load balancing schemes on heterogeneous networks. / In Proceedings of the Twelfth Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 30-38, 2000.
30. Diekmann, R., Frommer, A., and Monien, B. Efficient schemes for nearest neighbor load balancing. / Parallel Comput. V. 25, N. 7, pp. 789-812, 1999
31. Litow, B. The influence of graph structure on generalized dimension exchange. // Inf. Process. Lett. V. 54, N. 6, 347-353, 1995
32. Houle, M. E., Symvonis, A., and Wood, D. R. Dimension-exchange algorithms for token distribution on tree-connected architectures. / J. Parallel Distrib. Comput. V. 64, N. 5 , pp. 591-605, 2004
33. G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. // J. Parallel and Distr. Comp., V. 7, N. 2, pp. 279-301, 1989.
34. S. Hosseini, B. Litow, M. Malkawi, J. McPherson, K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. // J. Parallel and Distributed Comp., V. 10, N. 2, pp. 160-166, 1990
35. П. Ланкастер. Теория матриц. – М.:ФИЗМАТЛИТ, 1973.

Хританков Антон Сергеевич. Аспирант. Окончил Московский физико-технический институт в 2007 году. Имеет 12 печатных работ. Область научных интересов: оценка производительности вычислительных систем, распределенные вычисления, проектирование программных систем. E-mail: anton.khritankov@gmail.com.