Способ балансировки нагрузки при распределенном решении задач оптимизации методом ветвей и границ

Хританков А.С.
Московский Физико-Технический Институт
г. Долгопрудный, 141700, Россия, anton.khritankov@gmail.com

В работе предлагается подход к организации решения задач оптимизации методом ветвей и границ на вычислительной системе с распределенной памятью. Для моделирования процесса решения разработана автономная математическая модель, основанная на потоках подзадач в системе. В рамках предлагаемой модели ставится и решается задача распределения подзадач между узлами распределенной системы. Вычислительный эксперимент показал, что применение алгоритма позволяет добиться одинаковой загрузки узлов в процессе решения задачи оптимизации. Приводятся сравнительный анализ разработанного алгоритма и других алгоритмов балансировки нагрузки.

Введение

Пусть имеется задача оптимизации (в дальнейшем просто *задача*), решаемая одним из алгоритмов метода ветвей и границ. Общая схема метода ветвей и границ такова, что решение задачи связано с разделением исходной задачи на подзадачи, решаемые на части допустимой области. Последовательное применение разбиения задачи на подзадачи приводит к построению дерева подзадач, причем каждая из подзадач может быть решена независимо от других. Возможность независимого решения подзадач, как только они были определены, позволяет эффективно применять метод ветвей и границ при решении задач в параллельных и распределенных системах. Вследствие отсева подзадач, решение которых заведомо не приведет к получению оптимального решения, структура дерева подзадач не может быть определена до окончания решения. Для того чтобы загрузить имеющиеся вычислительные ресурсы на всем протяжении решения задачи, необходимо использовать динамические алгоритмы балансировки нагрузки.

В данной работе предлагается модель процесса вычислений и динамический алгоритм балансировки нагрузки, применение которого позволяет уменьшить время простоя узлов распределенной вычислительной системы по сравнению с другими алгоритмами. Разработанный алгоритм ориентирован на применение при решении задач с использованием систем, для которых время решения подзадачи сравнимо со временем, необходимым для передачи подзадачи от одного узла другому.

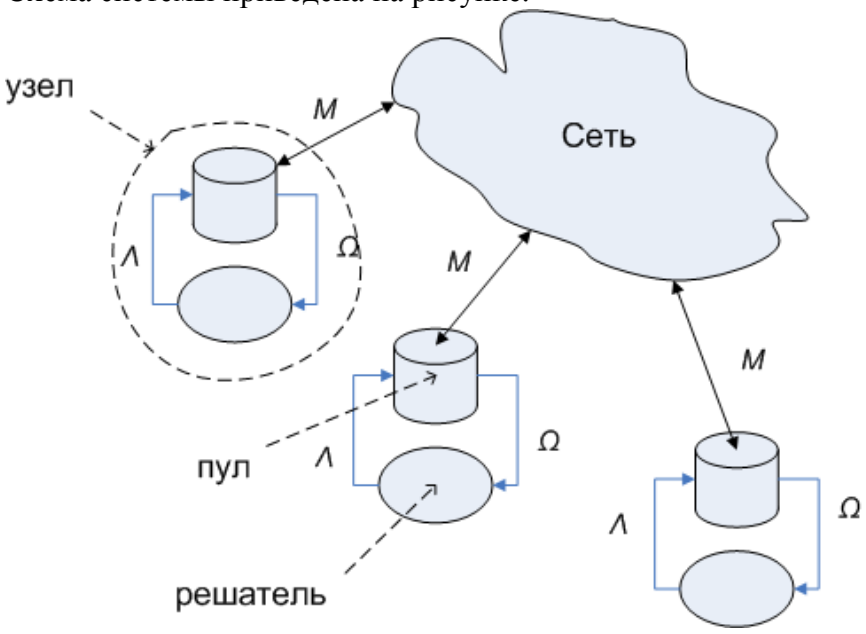
Подход

Совокупность вычислительных узлов, объединенных вычислительной сетью, в которой каждый узел может обмениваться данными с любым другим, будем называть *решающей системой*. Количество узлов в решающей системе и их характеристики могут изменяться в процессе решения.

Разработанный динамический алгоритм включает в себя использование специфической архитектуры решающей системы. В данной архитектуре определены два типа узлов: рабочий, который решает подзадачи, и управляющий, на котором расположена служба распределения нагрузки. Решающая система организована как одноранговая система, в которой все рабочие узлы равноценны. При увеличении числа узлов можно использовать иерархическую структуру, в которой присутствует несколько служб распределения нагрузки.

Каждый рабочий узел содержит *решатель*, реализующей алгоритм ветвей и границ и решающий подзадачи, и *пул подзадач*, используемый для временного хранения созданных и оцененных подзадач. Решатель использует комбинированный Best First Search – Depth First Search алгоритм позволяющий использовать ограниченный объем памяти пула и управлять числом создаваемых подзадач. Узлы могут обмениваться подзадачами между собой, процесс обмена подзадачами управляется службой распределения нагрузки, с которой узлы взаимодействуют в процессе решения задачи. Работа узла характеризуется тремя параметрами: общее число Λ созданных

подзадач, количество Ω решенных подзадач, а также число переданных в систему подзадач M. Схема системы приведена на рисунке.



На одном из узлов (ведущем узле) системы располагается исходная задача оптимизации, которую необходимо решить. Ведущий узел приступает к решению задачи, создавая новые подзадачи и обращаясь к службе балансировки за рекомендацией по передаче и распределению подзадач. Служба распределения нагрузки рассчитывает оптимальное распределение подзадач между узлами системы и возвращает рекомендованное значение числа подзадач, которые необходимо передать другим узлам.

Ведущий узел использует возможность управления алгоритмом решения и создает необходимое дополнительное количество подзадач, одновременно передавая их другим узлам в соответствии с рекомендациями службы балансировки. Другие узлы ожидают получения подзадач, после чего действуют по указанному выше алгоритму, пока в системе не останется подзадач для решения. Значение наилучшего решения (рекорда) распространяется через службу распределения нагрузки. *Процесс решения задачи моделируется при помощи трех моделей (в данной работе представлены две): квазистационарной динамической модели, потоковой статической и модели работы узла (работы комбинированного алгоритма).*

Критерии оценки

Естественным критерием оценки производительности системы является время решения поставленной задачи. Однако, при использовании метода ветвей и границ с несколькими решателями, время решения, а также число решенных подзадач, не является постоянным от прогона к прогону вследствие разного отсева. Воспользуемся другим критерием оценки производительности, эквивалентным упомянутому выше для случая детерминированного обхода дерева. В качестве *характеристики производительности* решающей системы используется число подзадач, решаемых всей системой в единицу времени. Для одинаковых путей обхода дерева поиска, большее число решаемых в единицу времени подзадач приводит к уменьшению общего времени решения.

Комбинированный алгоритм

Для управления количеством создаваемых подзадач, а также для использования ограниченного объема пула подзадач, предлагается использовать комбинированные алгоритмы, построенные с помощью объединения алгоритмов из двух классов.

Best First Search – это класс алгоритмов информированного поиска, основанных на выборе следующей подзадачи на основе некоторой эвристики. Выбор следующей подзадачи для решения происходит из множества всех узлов, для которых эвристика была рассчитана.

Depth First Search – это класс алгоритмов информированного поиска, включающих в себя два метода поиска – жадный алгоритм и откат (backtracking). Алгоритмы информированного поиска в глубину (Depth First Search) используют эвристические данные для оценки подзадач, полученных на предыдущем шаге, и выбирают подзадачу с лучшей оценкой, действуя подобно жадным алгоритмам. Оцененные подзадачи, которые не были выбраны, отбрасываются. Когда

жадный алгоритм достигает границы, либо все созданные подзадачи отсеиваются, производится откат, алгоритм возвращается к рассмотрению предыдущей подзадачи и выбирает следующую подзадачу.

Для решения подзадач, решатель применяет *комбинированный метод*. Суть комбинированного метода в том, что задается *решающее правило комбинирования алгоритмов*, на основе которого производится выбор, передавать ли следующую рассматриваемую подзадачу в пул подзадач вместо того, чтобы решать ее на следующем шаге алгоритма. Условно, применение решающего правила можно записать как $Метод = \theta \cdot DFS + (1-\theta) \cdot BFS$, где $\theta \in [0,1]$ - параметр выбора метода, определяемый решающим правилом

Квазистационарная модель

На основании экспериментальных данных, собранных при исследовании процесса решения задачи о ранце с помощью комбинированного Best First Search – Depth First Search алгоритма было установлено, что в процессе решения зависимость между Λ и Ω имеет случайный характер и зависит от пути обхода дерева поиска, выбора алгоритма решения и самой задачи. На рисунке приведена зависимость $\Lambda - M$ от времени.

Была выдвинута и экспериментально подтверждена *гипотеза, что зависимость $\Lambda - M$ от времени можно аппроксимировать ломаной* в метрике $\max\limits_{0 \le x \le T} |f(x) - g(x)|$, такой, что она будет иметь небольшое число звеньев, в зависимости от максимально допустимого отклонения δ, определяющего точки излома.

Значения производных по времени на линейных участках (*промежутках квазистационарности*) обозначим через

$\lambda = d\Lambda / dt, \ \omega = d\Omega / dt, \ \mu = d\mathrm{M} / dt$

Потоковая модель

Решающей системе на каждом промежутке квазистационарности сопоставляется полный граф и сеть с неограниченными пропускными способностями ребер. Потоком в сети будут передаваемые узлами подзадачи. Вершины графа сопоставляются узлам решающей системы, каждая вершина сети будет либо стоком (узел принимает подзадачи, $\mu < 0$), либо истоком (узел передает подзадачи, $\mu > 0$), либо нейтральной $\mu = 0$. Для сети выполняется закон сохранения потока $\sum\limits_{i \in U} \mu_i = 0$. Введем эвристическую характеристику загруженности узла $u_i = \dfrac{\lambda_i - \mu_i}{\omega_i}$

Аналогично для всей системы, ограничение сверху единицей необходимо для того, чтобы все истоки были генераторами, а стоки – обработчиками.

$$u_\Sigma = \min\left\{1, \frac{\sum\limits_{i \in U} \lambda_i}{\sum\limits_{i \in U} \omega_i}\right\} \ge 0$$

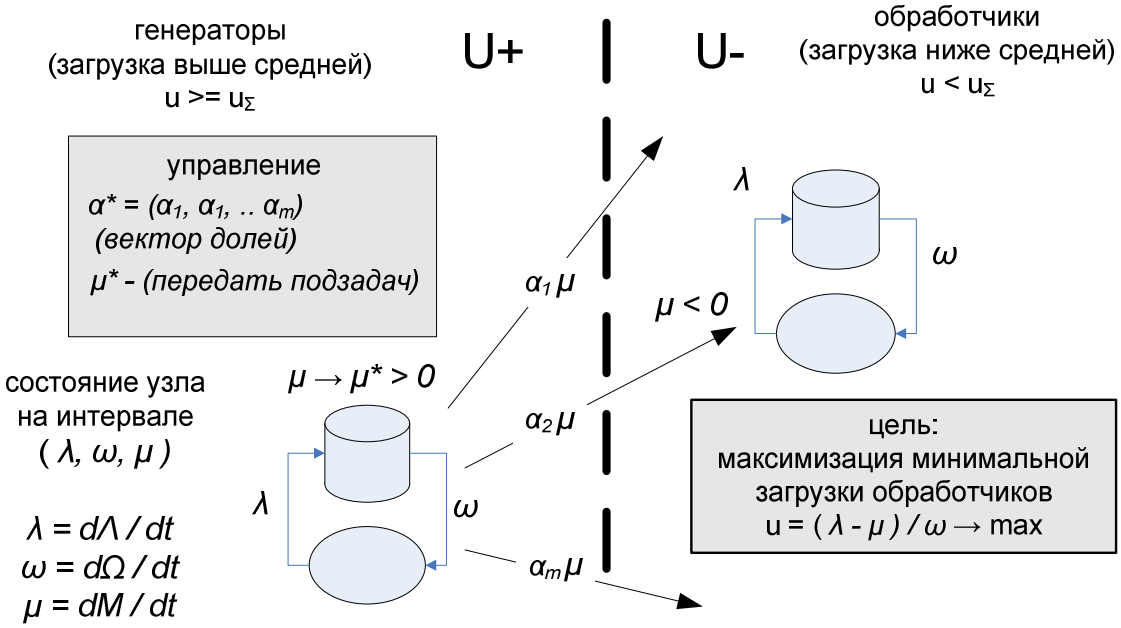
$U^+ = \{i \in U : u_i \ge u_\Sigma\}$ – *множество генераторов*

$U^- = \{i \in U : u_i < u_\Sigma\}$ – *множество обработчиков*

Задача распределения подзадач: $\mu_i^* = \alpha_i^* \mu \ (i \in U^-) \ \Rightarrow \ u_i^* = \dfrac{\lambda_i - \alpha_i^* \mu}{\omega_i}$

$$u^* \to \max_{(u^*,\alpha)\in A}, \quad A=\left\{(u^*,\alpha)\in \Re^{m+1} : \sum_{U^-}\alpha_i = 1,\ \alpha_i \ge 0;\ u^* \le u_i^* < u_\Sigma,\ i\in U^-\right\}$$

Схема потоковой модели приведена на рисунке.



Решение задачи распределения получено аналитически. Оптимальность решения доказана сведением задачи балансировки к общей задаче линейного программирования. Решение задачи:

$$\mu<0,\quad \rho_i=\frac{\lambda_i}{\omega_i} \qquad \boxed{u^*=\frac{\lambda-\mu}{\omega},\quad \alpha_i^*=\begin{cases}\dfrac{\omega_i}{\mu}\left(\rho_i-u^*\right), & \rho_i<u^*\\ 0. & \rho_i\ge u^*\end{cases}} \qquad u_i^*=\begin{cases}u^*, & \rho_i<u^*\\ \rho_i, & \rho_i\ge u^*\end{cases}$$

Для управления числом подзадач, создаваемых генераторами применяется механизм рекомендаций. Рекомендацией называется число подзадач, которое служба балансировки ожидала, что генератор передаст на интервале квазистационарности. Рекомендации μ_i^* $(i\in U^+)$ распределяются между генераторами пропорционально числу создаваемых подзадач, индекс U- означает суммарное значение по множеству U-

$$p_i^*=\frac{\lambda_i}{\sum_{i\in U^+}\lambda_i},\ S=\omega_{U^-}-\lambda_{U^-}+\mu,\ \Rightarrow \quad \boxed{\mu_i^*=p_i^*S=\frac{\lambda_i}{\sum_{i\in U^+}\lambda_i}S=\frac{\lambda_i S}{\lambda_{U^+}}},\quad \lambda_{U^+}=\sum_{i\in U^+}\lambda_i,\quad i\in U^+$$

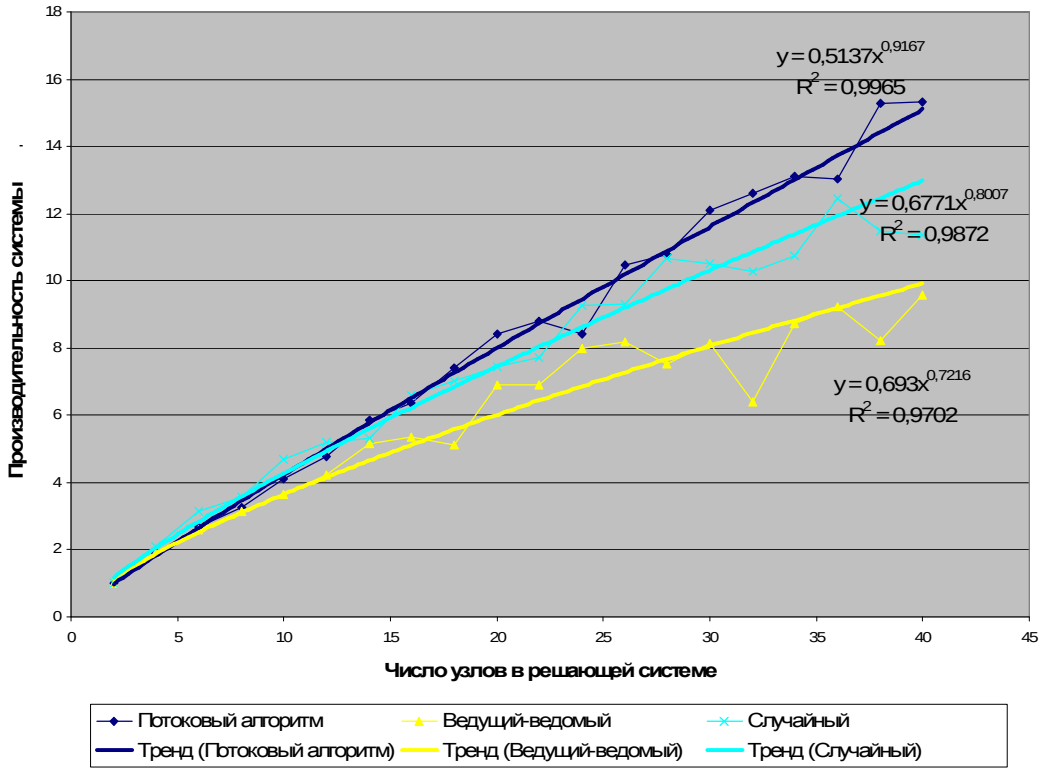
Эксперимент

Для проверки эффективности алгоритма была разработана имитационная программа. В программе для сравнительного анализа моделируется работа двух алгоритмов балансировки, кроме предложенного в настоящей работе:

- Алгоритм заимствования подзадач (Random Stealing), описанный в работе [2].
  Суть алгоритма состоит в заимствовании подзадач у случайно выбранного узла системы, когда локальный пул подзадач пуст. Для решения подзадач применяется комбинированный алгоритм.
- Алгоритм ведущий-ведомый (Master-Worker), предложенный в статье [1]
  Алгоритм перераспределяет подзадачи между узлами системы так, чтобы число подзадач на узле равнялось среднему по системе. Ведущий узел вычисляет среднее числа подзадач, опрашивая ведомые узлы через равные промежутки времени.

Под *производительностью* узла будем понимать число подзадач, решаемых узлом в единицу времени. Под производительностью решающей системы будем понимать количество подзадач, решаемых всеми узлами системы в единицу времени.

Анализ масштабируемости производится с помощью имитационной модели, в которой моделируется решение задачи плотным деревом поиска с размером $S = 10^7$ и максимальным порядком вершины N = 100. Размер пула подзадач одинаков для всех узлов и всех алгоритмов и равен B = 10000. По вертикальной оси отложена относительная производительность решающей системы. За единичную производительность принята производительность базовой системы, состоящей из двух узлов с относительными производительностями {1, 2}. Системы с большим числом узлов строятся добавлением базовой системы.



Видно, что для небольшого числа узлов (N < 10) алгоритмы одинаково эффективны, что связано с большим числом ветвей у вершин в дереве поиска в начале решения модельной задачи. Поэтому на всех узлах системы локальный пул подзадач остается заполненным достаточно долго, и решение происходит методом Depth First Search. При увеличении числа узлов в решающей системе возникает необходимость балансировки, так как среднее число узлов, в пуле которых недостаточно задач, растет. При 40 узлах предлагаемый алгоритм на 20% эффективнее алгоритма случайного выбора и почти в полтора раза эффективнее алгоритма ведущий-ведомый.

Библиография

[1] Kento Aida, Wataru Natsume, Yoshiaki Futakata, "Distributed Computing with Hierarchical Master-worker Paradigm for Parallel Branch and Bound Algorithm," Proc. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003), pp.156-163, May. 2003

[2] Blumofe, R. D. and C. E. Leiserson: 1994, ”Scheduling Multithreaded Computations by Work Stealing”. In: 35th Annual Symposium on Foundations of Computer Science (FOCS '94). Santa Fe, New Mexico, pp. 356-368

[3] Сигал И.Х., Иванова А.П. «Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы» Учеб. пособие. – Изд. 2-е. испр. – М.: ФИЗМАТЛИТ, 2003