Использование параллельной СУБД PargreSQL для интеллектуального анализа сверхбольших графов

15

Разбиение графа на подграфы представляет собой задачу интеллектуального анализа графов, решение которой играет важную роль в ряде теоретических и практических задач (раскраска графа, проектирование БИС и ПЛИС, конечно-элементное моделирование и др.). Существующие последовательные и параллельные алгоритмы предполагают возможность размещения графа и промежуточных данных обработки в оперативной памяти и неприменимы для случая сверхбольших графов. Описан подход к обработке сверхбольших графов на основе использования параллельной реляционной СУБД PargreSQL, разработанной на базе свободной СУБД PostgreSQL.

Ключевые слова: интеллектуальный анализ данных, разбиение графа, параллельная СУБД

АВТОРЫ:

К.С. Пан – программист отдела интеллектуального анализа данных и виртуализации Лаборатории суперкомпьютерного моделирования Национального исследовательского Южно-Уральского государственного университета
e-mail: kvapen@gmail.com
М.Л. Цымблер – канд. физ.-мат. наук, доцент, начальник отдела интеллектуального анализа данных и виртуализации Лаборатории суперкомпьютерного моделирования Национального исследовательского Южно-Уральского государственного университета
e-mail: zymbler@gmail.com

Под интеллектуальным анализом данных (Data Mining) понимают совокупность методов, алгоритмов и программного обеспечения для обнаружения в данных ранее неизвестных и практически полезных знаний, необходимых для принятия решений в различных сферах человеческой деятельности. Одна из актуальных областей приложения технологий Data Mining — задачи интеллектуального анализа сверхбольших графов (имеющих миллионы вершин и/или ребер), возникающие при моделировании сложных структур: химические соединения, белковые структуры, биологические и социальные сети, Web, потоки работ, XML документы и др.

Задача разбиения графов (graph partitioning) является одной из задач интеллектуального анализа графов и определяется следующим образом (рис. 1). Пусть имеется граф $G = (N, E)$, где N — множество вершин, E — множество взвешенных ребер, и целое положительное число p. Требуется найти разбиение исходного графа на непересекающиеся подграфы $N_1, N_2, \ldots, N_p$, такие, что сумма весов ребер, соединяющих подграфы, минимальна.

Эффективное разбиение графов имеет большое значение в ряде теоретических и практических задач. В качестве примера теоретических задач можно привести задачи раскраски графа, определения числа и состава компонент связности графа и представления графа в виде ярусно-параллельной формы. Примеры практических задач, в которых необходимо разбиение графа, — проектирование сложных электронных схем, БИС (больших интегральных схем) и ПЛИС (программируемых логических интегральных схем), проектирование топологии локальной сети, конечно-элементное моделировании и др.

Для разбиения сверхбольших графов применяют так называемое многоуровневое разбиение, поскольку традиционные алгоритмы требуют недопустимо много времени и/или оперативной памяти. Многоуровневое разбиение (multilevel partitioning) предполагает три стадии этого процесса (рис. 2). На первой стадии выполняется «огрубление» (coarsening) графа, т. е. уменьшение количества вершин и ребер в нем путем «склеивания» вместе сильно связанных вершин и удаления возникающих при этом петель. На второй стадии выполняется начальное разбиение, когда огрубленный граф подвергают анализу обычными алгоритмами и получают грубое разбиение.

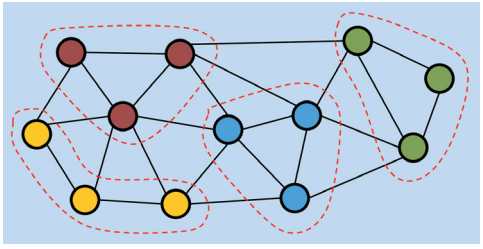

Рис. 1.
Задача разбиения графов

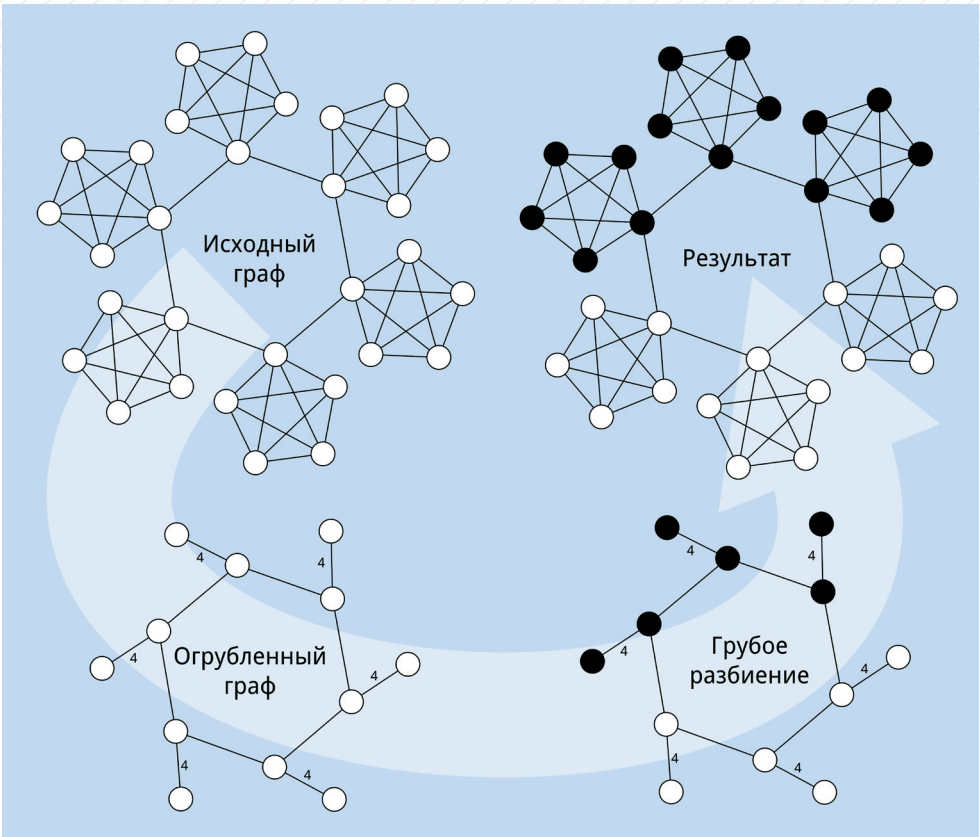

Рис. 2.
Многоуровневое разбиение графа

Затем, на третьей стадии, выполняется «уточнение» (uncoarsening) грубого разбиения с помощью какой-либо эвристики.

Частным случаем задачи разбиения является бисекция графа ($p = 2$, разбиение исходного графа на два подграфа). Разбиение на большее количество подграфов выполняется рекурсивно, когда каждый из найденных подграфов подвергается бисекции. Процесс бисекции (рис. 3) начинается с присваивания вершинам произвольного «цвета», которым обозначается принадлежность вершины к тому или иному подграфу. После этого производится подсчет показателя «выгоды» для каждой вершины: насколько выгодно изменить цвет этой вершины на противоположный.

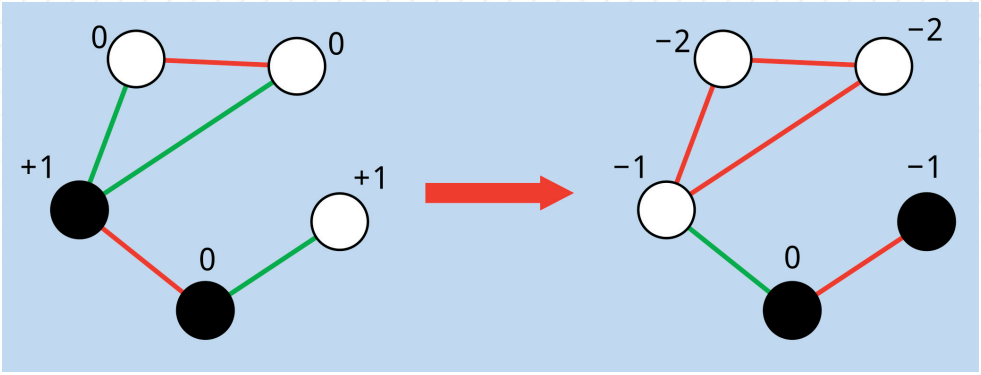

Рис. 3.
Бисекция графа

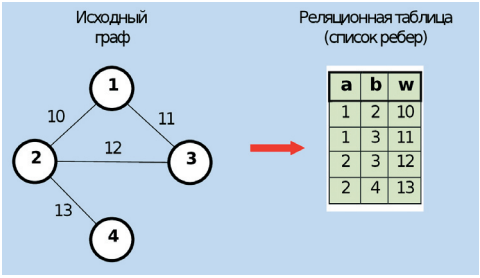

Рис. 4.
Представление графа в виде реляционной таблицы (список ребер)

Выгода равна сумме весов всех ребер, соединяющих данную вершину с другими вершинами. Если ребро соединяет вершины разного цвета, то его вес увеличивает сумму, иначе — уменьшает. Затем вершины, цвет которых оказалось выгодно изменить, подвергаются «перекрашиванию». Процесс повторяется до тех пор, пока такие вершины существуют.

Существующие в настоящее время системы, использующие последовательные и параллельные алгоритмы разбиения графов [2], предполагают размещение графов и промежуточных данных процесса обработки в оперативной памяти. Однако объем оперативной памяти современных компьютеров хотя и велик, но не беспределен. При обработке сверхбольших графов (имеющих миллионы вершин и/или ребер) упомянутые системы будут использовать жесткий диск для подкачки данных, что существенно замедлит решение задачи.

Реляционные системы управления базами данных (СУБД) на уровне программной архитектуры предполагают возможность эффективной реализации обработки данных, размер которых превышает объем доступной оперативной памяти. В соответствии с этим реляционные СУБД могут быть применены для обработки графов: реализация алгоритмов выполняется в виде набора SQL-

запросов, а базовыми структурами хранения и обработки данных выступают реляционные таблицы и индексы. Для представления графа в виде реляционной таблицы используется список ребер (рис. 4). Тем не менее в случае обработки графов сверхбольших размеров даже реляционная СУБД не может обеспечить надлежащую производительность.

Одним из решений данной проблемы является использование параллельной реляционной СУБД. При обработке запросов к реляционной базе данных параллельная СУБД использует концепцию фрагментного параллелизма (рис. 5). Реляционные таблицы подвергаются горизонтальной фрагментации по дискам кластерной системы. На каждом узле кластера устанавливается модифицированное ядро последовательной СУБД, которое воспринимает базу данных как набор «своих» фрагментов таблиц.

Модификация ядра последовательной СУБД заключается во внедрении в это ядро ряда механизмов параллельной обработки: автоматизированная фрагментация таблиц, распараллеливание запроса, координация действий ядер и обмены данными при выполнении запросов и др. Описанная идея реализована нами в параллельной СУБД PargreSQL [1], которая базируется на СУБД PostgreSQL, свободно распространяемой на уровне исходных кодов.

СУБД PargreSQL применена нами для решения задачи разбиения сверхбольших графов. Использование параллельной СУБД для решения данной задачи встречается, насколько нам известно, впервые. Схема решения задачи выглядит следующим образом (рис. 6). Сверхбольшой граф представляется в виде реляционной таблицы — списка ребер, распределя-

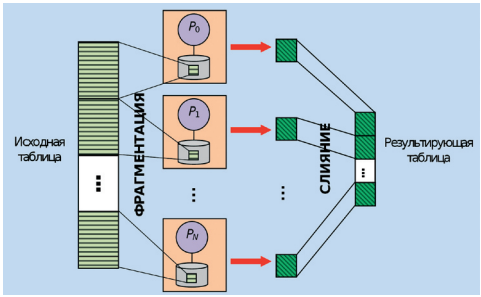

Рис. 5.
Параллельная обработка реляционных баз данных на основе фрагментного параллелизма

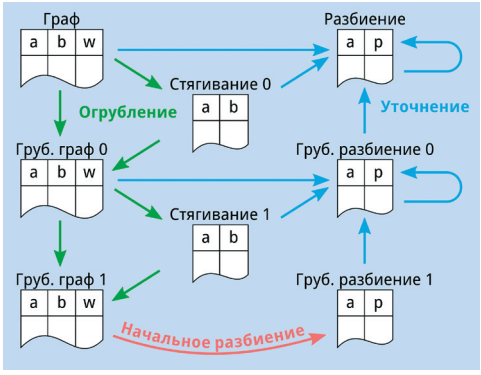

Рис. 6.
Схема многоуровневого разбиения графа с использованием СУБД

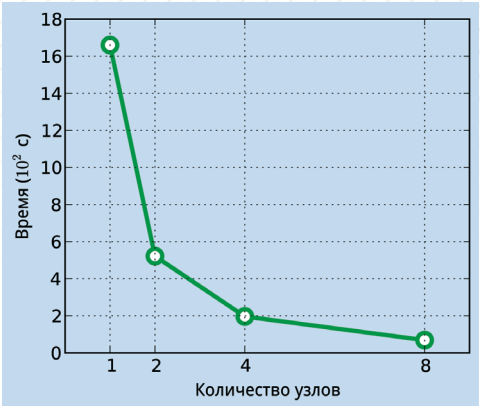

Рис. 7.
Результаты экспериментов

емой по узлам кластерной системы.

На стадии огрубления параллельная СУБД выполняет запрос на языке SQL, результатом которого является реляционная таблица со списком несмежных ребер, имеющих наибольший суммарный вес. После этого СУБД выполняет запрос, который осуществляет стягивание найденных ребер. Получаемые в ходе стягивания петли уничтожаются, а кратные ребра преобразуются в одно ребро, имеющее вес, равный сумме весов кратных ребер. Процесс повторяется многократно, пока граф не перестанет быть сверхбольшим.

Полученный огрубленный граф может быть размещен целиком в оперативной памяти, что позволяет перейти к стадии начального разбиения. Огрубленный граф экспортируется из базы данных и подается на вход сторонней утилиты, которая выполняет начальное разбиение с помощью одного из традиционных алгоритмов. Результат начального разбиения импортируется в базу данных в виде реляционной таблицы, содержащей список вершин графа с указанием их цвета.

На стадии уточнения параллельная СУБД выполняет SQL-запрос, который отменяет стягивание ребер графа и «окрашивает» концы восстановленных ребер в цвет соответствующей вершины огрубленного графа. После этого СУБД выполняет запрос, который находит более выгодное разбиение путем «перекрашивания» вершин согласно некоторой эвристике (рис. 3). Процесс уточнения повторяется столько раз, сколько итераций производилось на стадии огрубления. Итогом разбиения является реляционная таблица из двух столбцов: номер вершины графа и номер соответствующего подграфа.

Описанный подход был нами реализован, и были проведены вычислительные эксперименты, показавшие его эффективность (рис. 7).

В качестве аппаратной платформы экспериментов использовался суперкомпьютер «СКИФ-Аврора ЮУрГУ». В экспериментах выполнялась бисекция графа, который использовался в качестве исходных данных в конкурсе WISE Challenge международной конференции Web Information System Engineering

2012 [http://www.wise2012.cs.ucy.ac.cy]. В качестве инструмента начального разбиения графа использовалась свободно распространяемая утилита Chaco [3].

Исследование выполнено при финансовой поддержке РФФИ в рамках научных проектов №№ 12-07-00443-а и 12-07-31217 мол_а.

СПИСОК ЛИТЕРАТУРЫ

1. *Пан К.С., Цымблер М.Л.* Разработка параллельной СУБД на основе последовательной СУБД PostgreSQL с открытым исходным кодом // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». – 2012. – № 18(277). – Вып. 12. – С. 112–120.

2. *Aggarwal C.C., Wang H.* Managing and Mining Graph Data. Advances in Database Systems. – Vol. 40. – Springer, 2010. – 608 p.

3. *Hendrickson B. Chaco.* Encyclopedia of Parallel Computing. Springer, 2011. – P. 248–249.