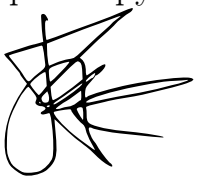
На правах рукописи



ПАН Константин Сергеевич

МЕТОДЫ ВНЕДРЕНИЯ ФРАГМЕНТНОГО ПАРАЛЛЕЛИЗМА В ПОСЛЕДОВАТЕЛЬНУЮ СУБД С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ

05.13.11 — математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата физико-математических наук

Челябинск — 2013

Работа выполнена на кафедре системного программирования
ФГБОУ ВПО «Южно-Уральский государственный университет»
(национальный исследовательский университет)

Научный руководитель: ЦЫМБЛЕР Михаил Леонидович
кандидат физ.-мат. наук, доцент
доцент кафедры системного программирования, ФГБОУ ВПО «Южно-Уральский государственный университет» (национальный исследовательский университет)

Официальные оппоненты: ЗЫКИН Сергей Владимирович
доктор техн. наук, профессор
зав. лабораторией методов преобразования и представления информации, ФГБУН «Институт математики им. С.П. Соболева» СО РАН;

СОБОЛЕВ Сергей Игоревич
кандидат физ.-мат. наук
научный сотрудник, Научно-исследовательский вычислительный центр ФГБОУ ВПО «Московский государственный университет имени М.В. Ломоносова»

Ведущая организация: ФГБУН «Институт программных систем имени А.К. Айламазяна» РАН

Защита состоится 18 декабря 2013 г. в 12 часов на заседании диссертационного совета Д 212.298.18 при Южно-Уральском государственном университете по адресу: 454080, г. Челябинск, пр. Ленина, 76, ауд. 1001.

С диссертацией можно ознакомиться в библиотеке Южно-Уральского государственного университета.

Автореферат разослан 15 ноября 2013 г.

Ученый секретарь
диссертационного совета  М.Л. Цымблер

Общая характеристика работы

Актуальность темы. В настоящее время одним из феноменов, оказывающих существенное влияние на область технологий обработки данных, являются *сверхбольшие данные*. В условиях современного информационного общества имеется широкий спектр приложений (социальные сети, электронные библиотеки, геоинформационные системы и др.), в каждом из которых производятся неструктурированные данные, имеющие сверхбольшие объемы и высокую скорость прироста (от 1 Терабайта в день). Исследования экспертов корпорации EMC показывают, что к 2020 г. мировой объем данных достигнет 40 Зеттабайт. Сверхбольшие данные путем интеллектуального анализа преобразуются в сверхбольшие реляционные базы данных, которые сохраняют в структурированном виде полученные результаты анализа, требующие параллельной обработки.

Сегодня *параллельные системы баз данных*, которые обеспечивают обработку запросов на многопроцессорных и многоядерных вычислительных системах, признаются научным сообществом как единственное эффективное средство для организации хранения и обработки сверхбольших баз данных. Базисной концепцией параллельных систем баз данных является *фрагментный параллелизм*, предполагающий разбиение отношений базы данных на горизонтальные фрагменты, которые могут обрабатываться независимо на разных узлах многопроцессорной системы.

Существующие сегодня коммерческие СУБД, использующие фрагментный параллелизм (Teradata, Greenplum, DB2 Parallel Edition и др.), имеют высокую стоимость и, во многих случаях, ориентированы на специфические аппаратно-программные платформы. Альтернативой коммерческим СУБД являются свободные СУБД с открытым исходным кодом. Однако на сегодня отсутствуют свободные СУБД, реализующие фрагментный параллелизм. В связи с этим перспективной является идея модернизации существующего исходного кода свободной последовательной СУБД для построения на ее основе параллельной СУБД для кластерных вычислительных систем путем *внедрения фрагментного параллелизма*. При этом модернизация исходного кода не должна быть масштабной. Полученная параллельная СУБД должна демонстрировать хорошую масштабируемость, тогда недостаток производительности (по сравнению с коммерческими СУБД) может быть преодолен добавлением в кластер новых вычислительных узлов с сохранением экономичности данного решения.

Таким образом, *актуальной* является задача разработки методов внедрения фрагментного параллелизма в свободные реляционные СУБД, позволяющих осуществить параллелизацию без масштабных изменений исходного кода.

Цель и задачи исследования. *Цель* данной работы состояла в разработке методов, архитектурных подходов и алгоритмов, обеспечивающих внедрение фрагментного параллелизма в имеющиеся последовательные СУБД, свободно распространяемые на уровне исходных кодов, а также в проверке разработанных методов путем их применения для решения задач аналитической и оперативной обработки сверхбольших баз данных. Для достижения этой цели необходимо было решить следующие *задачи*:

1. Разработать методы внедрения фрагментного параллелизма в последовательную СУБД с открытым исходным кодом.

2. На основе разработанных методов предложить архитектурные подходы и алгоритмы, реализующие фрагментный параллелизм в рамках последовательной СУБД с открытым исходным кодом. Выполнить параллелизацию одной из свободных последовательных СУБД.

3. Исследовать эффективность предложенных подходов применительно к решению задач классов OLAP и OLTP, связанных с обработкой сверхбольших баз данных.

Методы исследования. Проведенные в работе исследования базируются на реляционной модели данных. При разработке программной системы применялись методы объектно-ориентированного проектирования и язык UML. В разработке алгоритмов использован аппарат теории графов.

Научная новизна работы заключается в следующем:

1. Впервые разработаны эффективные методы внедрения фрагментного параллелизма в последовательную СУБД с открытым исходным кодом.

2. Впервые выполнено внедрение фрагментного параллелизма в последовательную СУБД PostgreSQL.

3. Разработан новый алгоритм разбиения сверхбольших графов, состоящих из миллионов вершин и ребер, ориентированный на реляционные СУБД с фрагментным параллелизмом.

Теоретическая ценность работы состоит в том, что в ней предложены методы, архитектурные решения и алгоритмы, позволяющие интегрировать фрагментный параллелизм в последовательные реляционные СУБД, свободно распространяемые на уровне исходных кодов.

Практическая ценность работы заключается в том, что путем применения предложенных методов к свободной СУБД PostgreSQL получена параллельная СУБД для кластерных систем, названная PargreSQL, которая применима для решения реальных задач, связанных с обработкой сверхбольших баз данных.

Апробация работы. Основные положения диссертационной работы, разработанные методы, алгоритмы и результаты вычислительных экспериментов докладывались на следующих научных конференциях:

— на международной научной конференции DEXA'2013 (The 24th International Conference on Database and Expert Systems Applications) (Чешская Республика, Прага, 26–30 августа 2013 г.);

— на международной научной конференции ADBIS'2013 (The 17th East-European Conference on Advances in Databases and Information Systems) (Италия, Генуя, 1–4 сентября 2013 г.);

— на международной научной конференции SYRCoDIS'2011 (The 7th Spring Researchers Colloquium on Databases and Information Systems) (Москва, 2–3 июня 2011 г.);

— на Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: все грани параллелизма» (Новороссийск, 23–28 сентября 2013 г.);

— на международной научной конференции «Параллельные вычислительные технологии (ПаВТ'2011)» (Москва, 28 марта – 1 апреля 2011 г.);

— на Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: экзафлопсное будущее» (Новороссийск, 19–24 сентября 2011 г.);

— на Втором Московском суперкомпьютерном форуме (МСКФ'2011), (Москва, 26–27 октября 2011 г.);

— на международной суперкомпьютерной конференции «Научный сервис в сети Интернет: суперкомпьютерные центры и задачи» (Новороссийск, 20–25 сентября 2010 г.).

Публикации. По теме диссертации опубликовано 11 печатных работ и получено одно свидетельство Роспатента об официальной регистрации программы для ЭВМ. Работы [1–4] опубликованы в изданиях, включенных ВАК в перечень журналов, в которых должны быть опубликованы основные результаты диссертаций на соискание ученой степени доктора и кандидата наук. В совместных работах научному руководителю М.Л. Цымблеру принадлежит постановка задачи, К.С. Пану принадлежат все полученные результаты.

Структура и объем работы Диссертация состоит из введения, четырех глав, заключения, библиографии и приложения. Объем диссертации составляет 101 страницу, объем библиографии — 98 наименований.

Содержание работы

Во введении приводится обоснование актуальности темы, формулируются цели работы, ее новизна и практическая значимость; приводится обзор работ по тематике исследования и кратко излагается содержание диссертации.

Первая глава, «Фрагментный параллелизм и СУБД с открытым кодом», содержит описание концепции фрагментного параллелизма и аналитический обзор современных последовательных СУБД, свободно распространяемых на уровне исходных кодов.

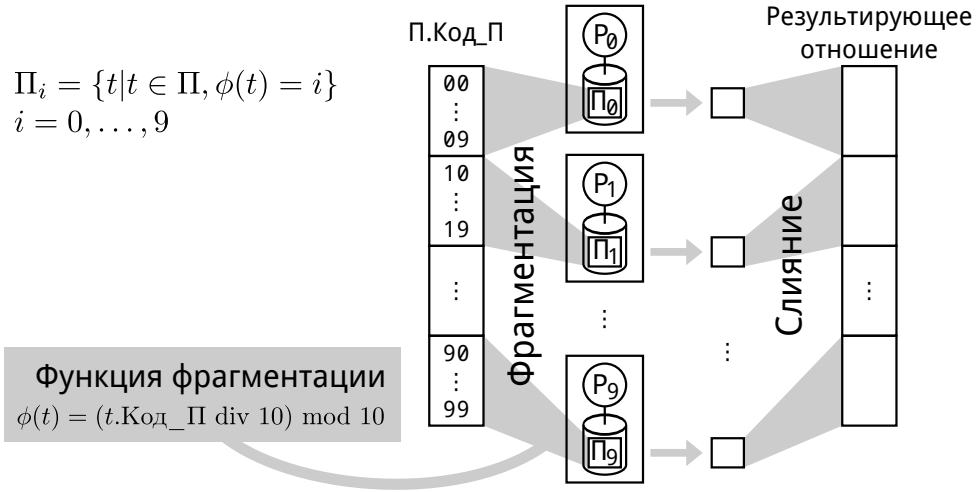


Рис. 1.

Фрагментный параллелизм (см. рис. 1) предполагает горизонтальную фрагментацию каждого отношения базы данных и распределение фрагментов по дискам многопроцессорной вычислительной системы. Способ фрагментации определяется *функцией фрагментации*, вычисляющей для каждого кортежа отношения номер процессорного узла, на котором должен быть размещен этот кортеж. Запрос параллельно выполняется на всех процессорных узлах в виде набора *параллельных агентов*, каждый из которых обрабатывает отдельный фрагмент отношения на выделенном ему процессорном узле. Полученные агентами результаты сливаются в результирующее отношение.

Функция фрагментации отношения R

$$\varphi : R \to \{0, 1, ..., k-1\}$$

вычисляет номер процессорного узла, на котором должен храниться кортеж. Величина k (количество процессорных узлов) называется *степенью*

фрагментации. Для фрагментации целесообразно использовать *атрибутную фрагментацию*, которая предполагает, что

$$\forall r \in R(\varphi_R(r) = f_A(r.A)),$$

где $f_A : D_A \to \{0, ..., k-1\}$ является некоторой функцией, определенной на домене атрибута A. То есть атрибутная фрагментация предполагает, что функция фрагментации зависит от определенного атрибута фрагментируемого отношения. Преимущество атрибутной фрагментации в том, что она допустима основными реляционными операциями: *группировка*, *удаление дубликатов*, *естественное соединение*.

Проведен сравнительный анализ существующих СУБД с открытым кодом с целью выбора СУБД, на примере которой предполагается применение разрабатываемых методов внедрения фрагментного параллелизма. Предложен набор критериев, на основе которых в качестве системы-кандидата для внедрения фрагментного параллелизма была выбрана свободная СУБД PostgreSQL. Описана архитектура СУБД PostgreSQL.

Во второй главе, «Внедрение фрагментного параллелизма в последовательную СУБД», предлагается комплекс методов для внедрения фрагментного параллелизма в последовательную СУБД с открытыми исходными кодами и описывается параллельная СУБД PargreSQL, реализованная путем внедрения фрагментного параллелизма в свободную СУБД PostgreSQL.

Внедрение фрагментного параллелизма осуществляется на основе следующих основных методов: тиражирование запроса, использование оператора обмена, построение параллельного плана запроса, обработка запросов на изменение данных, хранение метаданных о фрагментации отношений, портирование приложений последовательной СУБД и минимальная модификация исходных текстов последовательной СУБД.

Тиражирование запроса предполагает отправку данного запроса множеству экземпляров последовательной СУБД.

Оператор обмена exchange реализует пересылку кортежей между экземплярами СУБД во время выполнения запроса. Задача оператора *exchange* — передать все кортежи, которые должны быть обработаны экземплярами СУБД на других вычислительных узлах, и получить все кортежи, предназначенные для обработки экземпляром СУБД на данном вычислительном узле. Реализация оператора *exchange* инкапсулирует все аспекты, связанные с распараллеливанием запроса, так как он имеет стандартный итераторный интерфейс и может быть помещен в любое место плана запроса. Оригинальный исполнитель запросов последовательной СУБД выполняет этот оператор как и любой другой, не подразумевая никакого параллелизма.

Параллелизм достигается за счет построения параллельного плана запроса. Параллельный план запроса представляет собой последователь-

ный план запроса, в нужные места которого вставлены операторы *exchange* (см. рис. 2), обеспечивающие корректный результат при выполнении оригинальных алгоритмов исполнителя запросов.

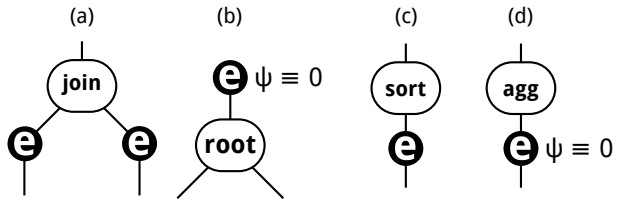


Рис. 2.

Обработка запросов на изменение данных (INSERT и UPDATE) обеспечивается посредством модификации плана запроса и оператора обмена. В случае INSERT в корень плана запроса добавляется дополнительная операция выборки, которая отсекает все кортежи, предназначенные для других вычислительных узлов. В случае UPDATE модифицированный алгоритм *exchange* позволяет перемещать кортежи, которые в результате обновления стали «чужими» (если $\varphi(t') \neq \varphi(t)$, то на узле с номером $\varphi(t)$ кортеж t удаляется, а на узле с номером $\varphi(t')$ вставляется кортеж t').

Хранение метаданных о фрагментации отношений осуществляется путем добавления в язык баз данных синтаксических средств определения функции фрагментации для таблиц, которая подлежит хранению в словаре базы данных.

Портирование приложений последовательной СУБД реализуется путем подключения новой прикладной библиотеки, которая имеет тот же интерфейс, что и оригинальная библиотека, и таким образом реализует прозрачное тиражирование запросов.

Модификация исходных текстов предполагает *внесение минимальных изменений*. Изменения в структурах данных и алгоритмах инкапсулируются в новых файлах исходных текстов, подключаемых к исходным текстам оригинальной СУБД.

Описанные методы *реализованы* в параллельной СУБД PargreSQL, архитектура которой показана на рис. 3. В рамках представленных методов последовательная СУБД рассматривается как подсистема параллельной СУБД. Методы предполагают как внесение изменений в подсистемы оригинальной СУБД, так и реализацию ряда новых подсистем.

Реализация оператора *exchange* предполагает внедрение в оригинальную СУБД PostgreSQL новых функций и типов данных. На рис. 4 представлен пакет *par_Exchange*, содержащий новые классы, добавляемые в СУБД PostgreSQL.

Классы данного пакета *Merge*, *Split*, *Scatter* и *Gather* реализуют одноименные узлы оператора *exchange*. Класс *Exchange_Builder* выполняет

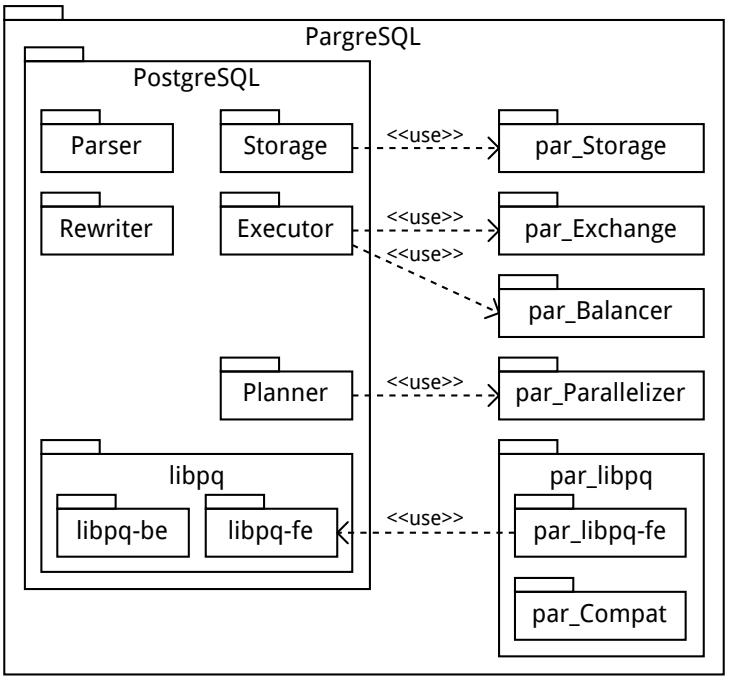


Рис. 3.

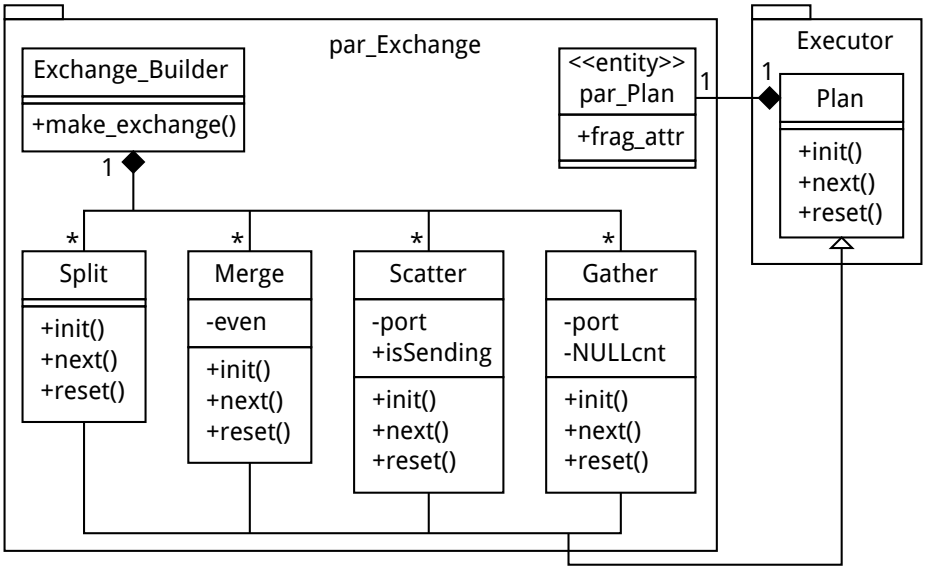


Рис. 4.

функции строителя, предоставляя метод для создания вышеперечисленных узлов плана и формирования из них цельного оператора *exchange*.

Для хранения атрибута фрагментации отношения необходимо выполнить модификацию класса *Plan* в СУБД PostgreSQL, который представляет собой узел плана запроса: в данный класс необходимо добавить целочисленный атрибут `frag_attr`.

В работе описаны алгоритмы реализации оператора обмена и методы его использования при выполнении запросов на выборку, изменение, вставку и удаление данных.

В третьей главе, «Применение параллельной СУБД Pargre-

SQL для интеллектуального анализа графов», представлен новый подход к решению задачи разбиения сверхбольших графов, которые состоят из миллионов вершин и ребер, ориентированный на реляционные СУБД с фрагментным параллелизмом. Данная задача решается с целью проверки применимости СУБД PargreSQL к задачам аналитической обработки сверхбольших баз данных. Описаны алгоритмы, которые реализуют стадии разбиения графа в виде запросов SQL.

Задача разбиения графов определяется следующим образом. Пусть имеется граф $G = (N, E)$, где N — множество взвешенных вершин, E — множество взвешенных ребер, и целое число $p > 0$. Требуется найти подмножества вершин N_1, N_2, ..., N_p из N такие, что

— $\cup_{i=1}^{p} N_i = N$ и $N_i \cap N_j = \varnothing$ для $i \neq j$;

— $W(i) \approx W/p, i = 1, 2, \ldots, p$, где $W(i)$ и W представляют собой суммы весов вершин N_i и N соответственно;

— величина *разреза (cut size)*, т.е. сумма весов ребер, соединяющих вершины разных подмножеств, минимальна.

Предложенное решение позволяет преодолеть трудности, связанные с ограниченным размером доступной оперативной памяти для хранения графа и промежуточных данных алгоритма, с которыми сталкиваются существующие алгоритмы интеллектуального анализа сверхбольших графов.

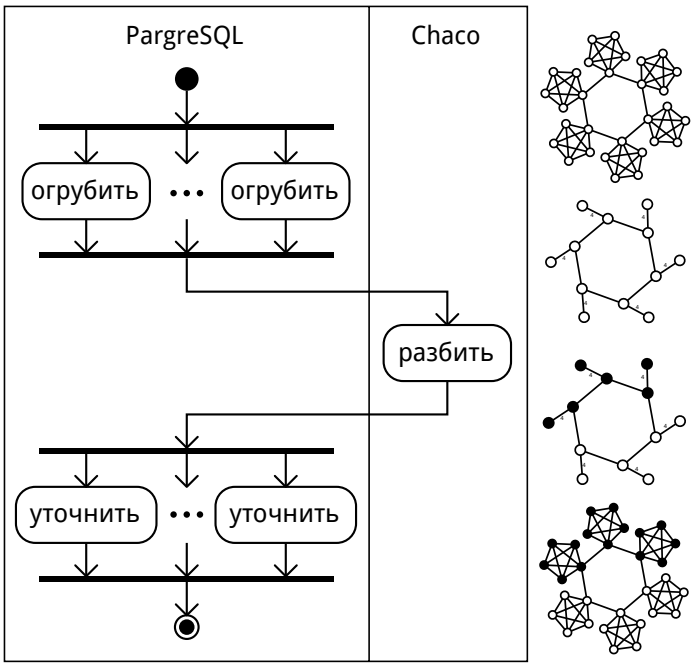


Рис. 5.

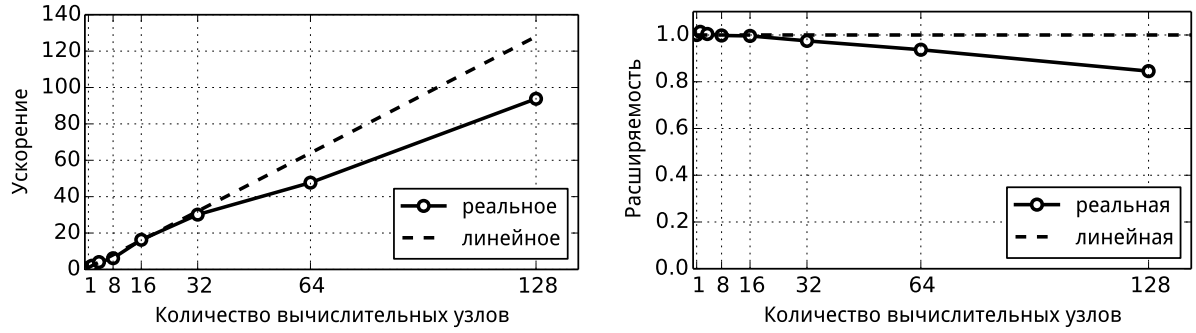
Разработанный алгоритм предполагает представление графа в виде реляционной таблицы (списка ребер), распределяемой по узлам кластерной системы. Разбиение состоит из трех последовательно выполняемых

стадий (см. рис. 5): огрубление, начальное разбиение и уточнение. Стадия огрубления выполняется с помощью СУБД PargreSQL и обеспечивает редукцию исходного графа до размеров, позволяющих разместить граф и промежуточные данные в оперативной памяти. На стадии начального разбиения огрубленный граф экспортируется из базы данных и подается на вход сторонней свободной утилиты Chaco, которая может выполнить начальное разбиение огрубленного графа в оперативной памяти с помощью одного из известных алгоритмов. Результат начального разбиения импортируется в базу данных в виде реляционной таблицы. На стадии уточнения разбиения параллельная СУБД PargreSQL с помощью запросов к таблице, полученной на предыдущей стадии, формирует финальное разбиение графа в виде реляционной таблицы из двух столбцов: номер вершины графа и номер подграфа, которому принадлежит эта вершина.

В четвертой главе, «Вычислительные эксперименты», приводятся результаты вычислительных экспериментов, выполненных на суперкомпьютере «Торнадо ЮУрГУ» с использованием разработанной параллельной СУБД PargreSQL. В рамках экспериментов исследованы ускорение и расширяемость СУБД PargreSQL, ее производительность на стандартных тестах консорциума TPC (Transaction Processing Council), а также эффективность и качество разбиения реальных сверхбольших графов.

В первой серии экспериментов исследовались основные показатели масштабируемости параллельной СУБД: ускорение и расширяемость. В данных экспериментах система PargreSQL исполняла запрос, предполагающий выполнение операции естественного соединения двух отношений по общему атрибуту.

В экспериментах на исследование ускорения соединяемые отношения имели размеры 300 млн. и 7.5 млн. кортежей соответственно, распределяемые по узлам кластера. Эксперименты показали (см. рис. 6a), что СУБД PargreSQL демонстрирует *ускорение, близкое к линейному.*



(a) Ускорение СУБД PargreSQL

(b) Расширяемость СУБД PargreSQL

Рис. 6.

В экспериментах, исследовавших расширяемость, размеры соединя-

емых отношений увеличивались пропорционально увеличению количества используемых узлов кластера с множителем 12 млн. и 0.3 млн. кортежей соответственно. В ходе экспериментов установлено (см. рис. 6b), что СУБД PargreSQL имеет *расширяемость, близкую к линейной.*

Вторая серия экспериментов была направлена на исследование эффективности СУБД PargreSQL на стандартных тестах производительности TPC-C. Тест TPC-C измеряет производительность СУБД на последовательности OLTP-запросов, моделирующих складской учет. В тесте использовалось от 1 до 30 параллельно работающих клиентов, выполняющих запросы к СУБД PargreSQL, запущенной на 12 узлах. Размер базы данных составлял 12 «складов».

Табл. 1.

К-во клиентов	tpm-C ↓	К-во клиентов	tpm-C ↓	К-во клиентов	tpm-C ↓	К-во клиентов	tpm-C ↓
29	2202531	24	2165413	16	1882353	8	1156626
26	2197183	23	2156250	15	1747572	7	1150684
30	2195122	22	2146341	14	1647058	5	857142
32	2194285	20	2068965	13	1529411	6	847058
27	2189189	19	2054054	12	1358490	4	657534
31	2188235	18	2037735	11	1346938	3	444444
28	2181818	21	2016000	10	1290322	2	328767
25	2173913	17	1961538	9	1270588	1	150000

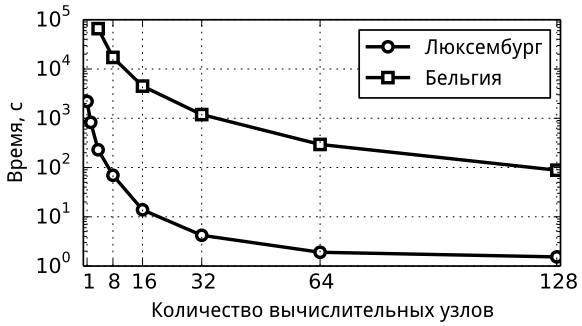
Результаты исследования эффективности СУБД PargreSQL на тесте TPC-C приведены в табл. 1 в порядке убывания показателя tpm-C (количество выполненных транзакций в минуту). Данный результат позволяет СУБД PargreSQL попасть в пятерку лидеров рейтинга TPC-C среди параллельных СУБД для кластеров (см. табл. 2).

В третьей серии экспериментов изучались ускорение и качество разбиения реальных сверхбольших графов, выполненного с помощью СУБД PargreSQL. В данных экспериментах выполнялось разбиение двух графов, представляющих собой карты дорог Люксембурга и Бельгии и содержащих 10^5 и 10^6 вершин соответственно. Эксперименты показали (см. рис. 7), что разбиение выполняется с ускорением, существенно превышающим линейное. Это является ожидаемым результатом, поскольку предложенное решение задачи относится к классу *embarrassingly parallel* («ошеломляюще параллельных»). При этом, однако, имеет место приемлемое качество разбиения исследуемых графов (см. рис. 8): не более 1.5% и не более 15% от общего количества вершин графа соответственно были причислены к неверным подграфам.

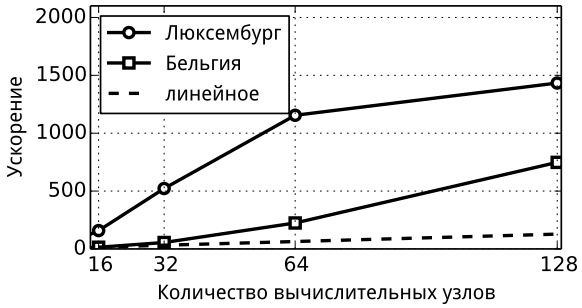
Качество разбиения графов оценивалось с помощью метрики Кернигана—Лина

Табл. 2.

№ п/п	Кластер	СУБД	К-во узлов	К-во клиентов	tpm-C
1	SPARC SuperCluster with T3-4 Servers	Oracle Database 11g R2 Enterprise Edition w/RAC w/Partitioning	108	81	30 249 688
2	IBM Power 780 Server Model 9179-MHB	IBM DB2 9.7	24	96	10 366 254
3	Sun SPARC Enterprise T5440 Server Cluster	Oracle Database 11g Enterprise Edition w/RAC w/Partitioning	48	24	7 646 486
	Торнадо ЮУрГУ	PargreSQL	12	29	2 202 531
4	HP Integrity rx5670 Cluster Itanium2/1.5 GHz-64p	Oracle Database 10g Enterprise Edition	64	80	1 184 893

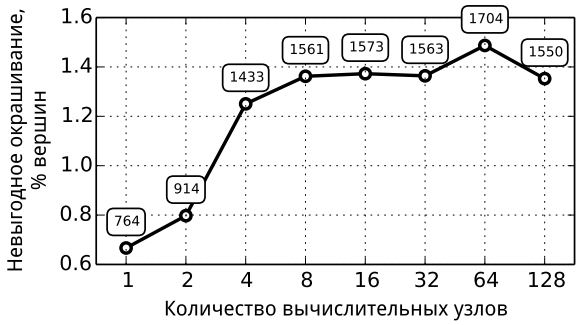


(a) Время разбиения графов

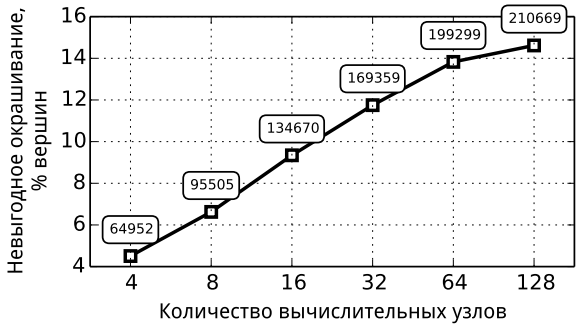


(b) Ускорение разбиения графов

Рис. 7.



(a) Качество разбиения графа Люксембурга



(b) Качество разбиения графа Бельгии

Рис. 8.

$$\text{gain}(v) = \sum_{v,u \in E, P(v) \neq P(u)} W(u,v) - \sum_{v,u \in E, P(v) = P(u)} W(u,v),$$

которая вычисляет для вершины v потенциальную выгоду от перемещения ее в другой подграф.

Таким образом, эксперименты показывают *хорошую масштабируемость* параллельной СУБД PargreSQL и *производительность на уровне*

зарубежных коммерческих аналогов, а также *успешное применение* PargreSQL для решения задачи разбиения сверхбольших графов.

В заключении суммируются основные результаты диссертационной работы, выносимые на защиту, приводятся данные о публикациях и апробациях автора по теме диссертации и рассматриваются направления дальнейших исследований в данной области.

В приложение вынесены статистические данные о популярности современных свободных СУБД с открытым исходным кодом, которые были использованы при принятии решения о выборе СУБД для внедрения фрагментного параллелизма.

Основные результаты диссертационной работы

На защиту выносятся следующие новые научные результаты:

1. Разработаны новые методы внедрения фрагментного параллелизма в свободную СУБД с открытым исходным кодом.

2. На основе разработанных методов предложены архитектурные подходы и алгоритмы, реализующие фрагментный параллелизм в рамках последовательной СУБД с открытым исходным кодом, на базе которых выполнена параллелизация СУБД PostgreSQL.

3. Разработан новый алгоритм разбиения сверхбольших графов, состоящих из миллионов вершин и ребер, ориентированный на реляционные СУБД с фрагментным параллелизмом.

4. Проведены вычислительные эксперименты с СУБД PargreSQL, которые показали успешное применение данной СУБД для решения задач классов OLAP и OLTP, связанных с обработкой сверхбольших баз данных.

Публикации по теме диссертации

Публикации в журналах из списка ВАК

1. *Пан К.С.* Разработка параллельной СУБД на основе PostgreSQL // Труды Института системного программирования РАН. 2011. Т. 21. С. 357–370.

2. *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.

3. *Pan C., Zymbler M.* Taming Elephants, or How to Embed Parallelism into PostgreSQL // Database and Expert Systems Applications – 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26–29, 2013. Proceedings, Part I. Springer, 2013. Lecture Notes in Computer Science. Vol. 8055. P. 153–164.

4. *Pan C., Zymbler M.* Very Large Graph Partitioning by Means of Parallel DBMS // Proceedings of the 17th East-European Conference on Advances in Databases and Information Systems, ADBIS'2013, September 1–4, 2013, Genoa, Italy. Lecture Notes in Computer Science. Springer, 2013. Vol. 8133. P. 388–399.

Другие публикации

5. *Pan C.* Development of a Parallel DBMS on the Basis of PostgreSQL // Proceedings of the Seventh Spring Researchers' Colloquium on Databases and Information Systems (SYRCoDIS'2011). Moscow: Moscow State University, 2011. P. 57–61.

6. *Пан К.С.* Подход к разбиению сверхбольших графов с помощью параллельных СУБД // Вестник ЮУрГУ. Серия «Вычислительная математика и информатика». 2012. № 47(306). Вып. 2. С. 127–132.

7. *Пан К.С.* Разработка параллельной СУБД на основе свободной СУБД PostgreSQL // Научный сервис в сети Интернет: экзафлопсное будущее: Труды Международной суперкомпьютерной конференции (Новороссийск, 19–24 сентября 2011 г.). М.: Изд-во МГУ, 2011. С. 566–571.

8. *Пан К.С., Цымблер М.Л.* Архитектура и принципы реализации параллельной СУБД PargreSQL // Параллельные вычислительные технологии (ПаВТ'2011): труды международной научной конференции (Москва, 28 марта – 1 апреля 2011 г.). Челябинск: Издательский центр ЮУрГУ, 2011. С. 577–584.

9. *Пан К.С., Цымблер М.Л.* Исследование эффективности параллельной СУБД PargreSQL // Научный сервис в сети Интернет: все грани параллелизма: Труды Международной суперкомпьютерной конференции (Новороссийск, 23–28 сентября 2013 г.). М.: Изд-во МГУ, 2013. С. 148–149.

10. *Пан К.С., Цымблер М.Л.* Проект PargreSQL: разработка параллельной СУБД на основе свободной СУБД PostgreSQL // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи: Труды Международной суперкомпьютерной конференции (Новороссийск, 20–25 сентября 2010 г.). М.: Изд-во МГУ, 2010. С. 308–313.

11. *Пан К.С., Цымблер М.Л.* Использование параллельной СУБД PargreSQL для интеллектуального анализа сверхбольших графов // Суперкомпьютерные технологии в науке, образовании и промышленности. 2012. № 1. С. 125–134.

Свидетельство о регистрации программы

12. *Соколинский Л.Б., Цымблер М.Л., Пан К.С., Медведев А.А.* Свидетельство Роспатента о государственной регистрации программы для ЭВМ «Параллельная СУБД PargreSQL» № 2012614599 от 23.05.2012.

Работа выполнялась при поддержке *Российского фонда фундаментальных исследований* (проекты 12-07-31217 мол_а, 12-07-00443, 09-07-00241) и *Президента Российской Федерации* (стипендия СП-5427.2013.5).