

Интеграция параллелизма в СУБД с открытым кодом

Пан К. С., Соколинский Л. Б. (sokolinsky@gmail.com), Цымблер М. Л. — ЮУрГУ (Челябинск)

В настоящее время СУБД с открытым исходным кодом (например, PostgreSQL, MySQL, SQLite и др.) являются надежной альтернативой коммерческим СУБД. PostgreSQL представляет собой одну из наиболее популярных СУБД с открытым кодом. Проект PostgreSQL был начат в 1995 году как ответвление от проекта POSTGRES М. Стоунбрейкера и до сих пор разрабатывается группой энтузиастов. Сначала PostgreSQL отличался от своего предка только наличием SQL-синтаксиса в запросах. На сегодня PostgreSQL представляет собой полноценную объектно-реляционную СУБД с открытым кодом для практически всех популярных операционных систем. PostgreSQL поддерживает стандарт SQL:2011, ACID-транзакции и хранимые процедуры на различных языках высокого уровня. Максимальный размер таблицы в PostgreSQL равен 32 Тбайт, максимальный размер поля таблицы — 1 Гбайт. PostgreSQL имеет хорошо документированный код и применяется в многочисленных коммерческих организациях, госструктурах и университетах (например, Apple, Sun, Cisco, Fujitsu, Red Hat, U.S. State Department, United Nations Industrial Development Organization и др.).

Проект PargreSQL посвящен разработке параллельной СУБД PargreSQL путем внедрения фрагментного параллелизма в СУБД PostgreSQL.

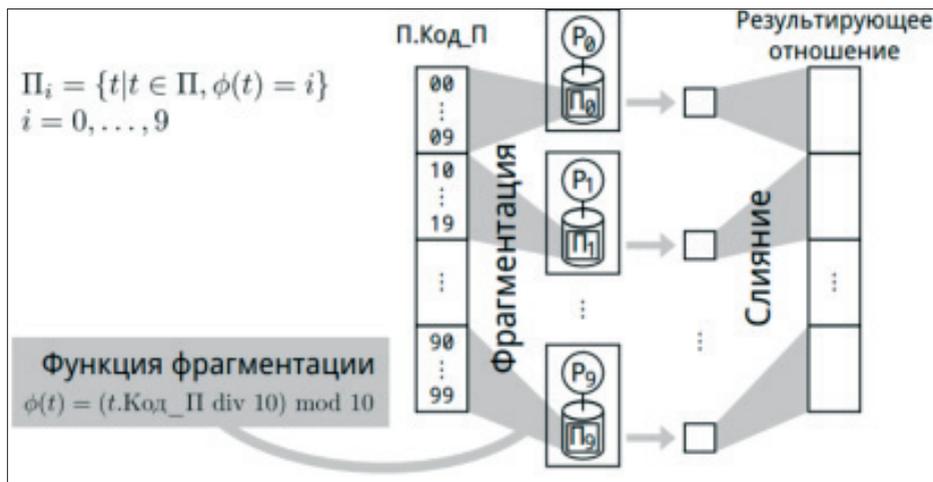


Рис. 1. Фрагментный параллелизм

Фрагментный параллелизм (см. рис. 1) подразумевает горизонтальную фрагментацию каждой таблицы базы данных по дискам кластерной системы. Способ фрагментации определяется функцией фрагментации, которая получает значение некоторой колонки таблицы и выдает номер диска, где хранится данная запись. На каждом узле запускается параллельный агент, представляющий собой модифицированный экземпляр СУБД PostgreSQL, который обрабатывает свои фрагменты, и затем частичные результаты сливаются в результирующую таблицу.

Технология внедрения параллелизма кратко может быть описана следующим образом. При обработке запроса мы добавляем к стандартным шагам (разбор запроса, разрешение представлений, построение плана запроса, выполнение плана запроса) еще два: построение параллельного плана запроса из последовательного плана и балансировка нагрузки узлов кластера при выполнении запроса (см. рис. 2).

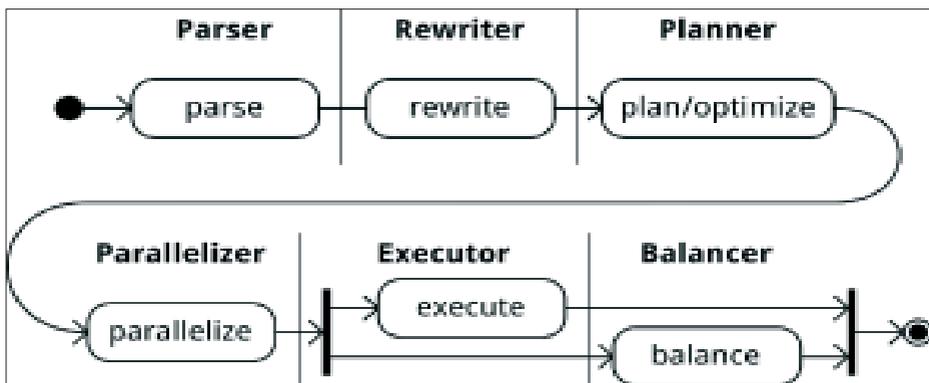


Рис. 2. Этапы обработки запроса в СУБД PostgreSQL

Для реализации фрагментации в синтаксис команды PostgreSQL создания таблиц нами добавлен дополнительный параметр, специфицируя который программист при создании таблицы указывает имя целочисленного поля, используемого для фрагментации. В качестве функции фрагментации берется остаток от деления поля фрагментации на количество вычислительных узлов в кластере.

Нами разработан набор макросов, который подменяет вызовы оригинальных функций PostgreSQL на их PargreSQL-копии и обеспечивает прозрачный переход последовательных приложений PostgreSQL на параллельные приложения PargreSQL.

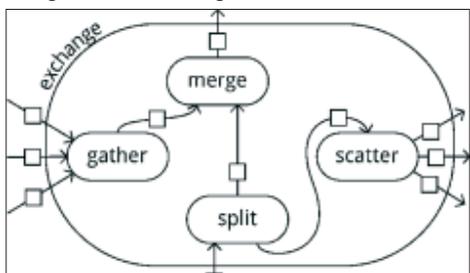


Рис. 3. Оператор обмена EXCHANGE

Оператор EXCHANGE (см. рис. 3) состоит из четырех операторов: Split, Scatter, Gather и Merge. Split разделяет записи на «свои» (они должны быть обработаны на текущем узле кластера) и «чужие» (их необходимо передать на другой узел). Scatter отправляет «чужие» записи на соответствующие им узлы. Gather принимает «свои» записи от других узлов. Merge попеременно выдает результаты Gather и Split. Оператор обмена EXCHANGE встав-

ляется в нужные места последовательного плана запроса и реализует пересылки данных между параллельными агентами, необходимые для обеспечения корректности результата запроса.

Нами проведены эксперименты по исследованию ускорения и расширяемости СУБД PargreSQL, а также ее эффективности на задачах класса OLTP (обработки транзакций). Эксперименты проводились на суперкомпьютере «Торнадо ЮУрГУ», который занял 249-е место в 41-й редакции рейтинга TOP500 (июнь 2013 года).

В экспериментах на исследование ускорения выполнялся запрос, предполагающий соединение двух таблиц размерами 3×10^8 и $7,5 \times 10^5$ записей соответственно. Нами получено ускорение, близкое к линейному (см. рис. 4).

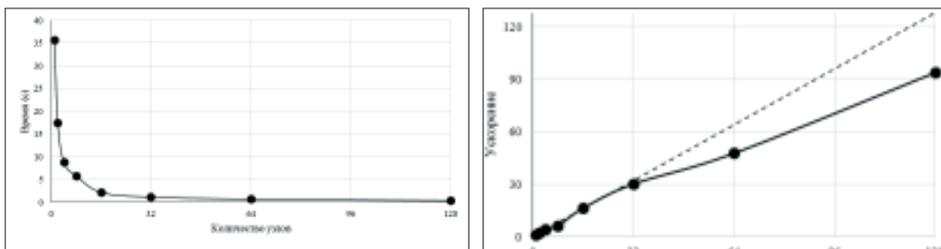


Рис. 4. Результаты экспериментов по исследованию ускорения

На том же запросе нами была исследована расширяемость СУБД PargreSQL с использованием от 1 до 128 узлов, когда одновременно с увеличением количества узлов равно увеличивается объем данных (от $1,2 \times 10^6$ и 3×10^5 записей до $1,5 \times 10^8$ и $3,8 \times 10^6$ записей соответственно). Эксперименты показали (см. рис. 5), что расширяемость близка к линейной.

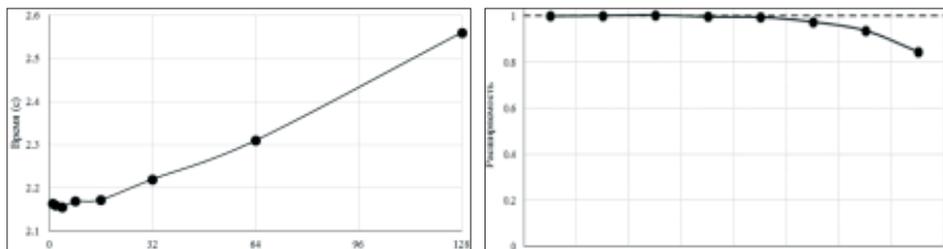


Рис. 5. Результаты экспериментов по исследованию расширяемости

Эффективность СУБД PargreSQL на задачах OLTP измерялась на стандартном тесте TPC C консорциума TPC (Transaction Processing Council), в котором моделируется складской учет. На конфигурации «12 складов, 30 клиентов» с использованием 12 узлов кластера PargreSQL показал производительность 2,2 млн запросов в минуту.