УДК 004.65

РАЗРАБОТКА ПАРАЛЛЕЛЬНОЙ СУБД НА ОСНОВЕ ПОСЛЕДОВАТЕЛЬНОЙ СУБД POSTGRESQL С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ

К.С. Пан, М.Л. Цымблер

Статья посвящена разработке параллельной системы управления базами данных (СУБД) путем внедрения механизмов параллельной обработки запросов на основе концепции фрагментного параллелизма в свободно распространяемую на уровне исходных кодов СУБД PostgreSQL. Описана архитектура и принципы реализации параллельной СУБД PargreSQL для кластерных вычислительных систем, разрабатываемой на основе свободно распространяемой СУБД PostgreSQL. СУБД PostgreSQL является подсистемой в рамках системы PargreSQL. Описаны изменения, которые требуется внести в исходные тексты подсистем СУБД PostgreSQL. В исходные тексты PostgreSQL вносятся минимальные изменения. Изменения в структурах данных и алгоритмах инкапсулируются в новых файлах исходных текстов, подключаемых к исходным текстам PostgreSQL. Использование PargreSQL является прозрачным для пользовательских приложений. Подключение PargreSQL к прикладным программам, которые до этого использовали PostgreSQL, производится с минимальными изменениями в исходных кодах приложения. Параллельная СУБД PargreSQL, запущенная на одном вычислительном узле, работает так же, как последовательная СУБД PostgreSQL.

Ключевые слова: параллельные СУБД, фрагментный параллелизм, PostgreSQL.

Введение

СУБД PostgreSQL [1] представляет собой свободно распространяемую реляционную СУБД с открытым исходным кодом. Научный проект Омега [2] направлен на разработку прототипа параллельной СУБД для мультипроцессорных вычислительных систем с кластерной архитектурой.

Параллельная СУБД PargreSQL разрабатывается в рамках проекта Омега. Базовой идеей этой разработки является внедрение поддержки фрагментного параллелизма [3] в СУБД PostgreSQL. В данной работе описана архитектура и основные принципы разработки СУБД PargreSQL.

1. Работы по тематике исследования

В настоящее время СУБД PostgreSQL представляет собой надежную свободно распространяемую на уровне исходных кодов альтернативу коммерческим СУБД. Существует достаточно большое количество практических приложений баз данных на основе PostgreSQL и исследовательских проектов, посвященных расширению и улучшению PostgreSQL.

В работе [4] рассматривается внедрение поддержки XML в PostgreSQL. Добавление новых типов данных для обеспечения поддержки стандарта обмена медицинской информацей HL7 в PostgreSQL описывается в [5]. Авторы работы [6] предлагают расширение PostgreSQL для обработки изображений. В работе [7] представлен подход к интеграции PostgreSQL с Semantic Web.

Исследования, посвященные использованию PostgreSQL для параллельной обработки запросов, могут быть представлены следующими работами. В [8] предлагается расширение PostgreSQL для распределенной обработки запросов. Описаны необходимые изменения

в исполнителе запросов PostgreSQL и предложены соответствующие способы увеличения производительности.

СУБД ParGRES [9] представляет собой промежуточное программное обеспечение (middleware) с открытым исходным кодом для высокопроизводительной обработки OLAP-запросов. ParGRES использует внутризапросный параллелизм на кластерных вычислительных системах и адаптивное виртуальное распределение базы данных. СУБД GParGRES [10] является продолжением продукта ParGRES для грид-сред. GParGRES использует репликацию базы данных и меж- и внутризапросный параллелизм для эффективной обработки OLAP-запросов в грид. Предложенный подход подразумевает разбиение данных на двух уровнях: на уровне грид (реализовано в GParGRES) и на уровне узлов (реализовано в ParGRES).

Нами предлагается внедрение фрагментного параллелизма [11] в СУБД PostgreSQL на основе методов параллельной обработки запросов, разработанных в рамках проекта Омега [12, 13].

2. Архитектура PostgreSQL

В основе архитектуры PostgreSQL лежит модель «клиент-сервер». В сеансе работы с PostgreSQL участвуют три вида взаимодействующих процессов (см. рис. 1): *приложение-клиент (frontend)*, *серверный процесс (backend)* и *демон (daemon)*. Демон осуществляет прием соединений, устанавливаемых клиентами, и создает отдельный серверный процесс для обработки запросов каждого отдельного клиента.

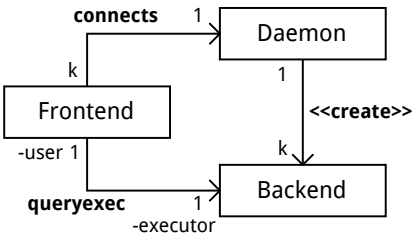

Рис. 1. Процессы СУБД PostgreSQL

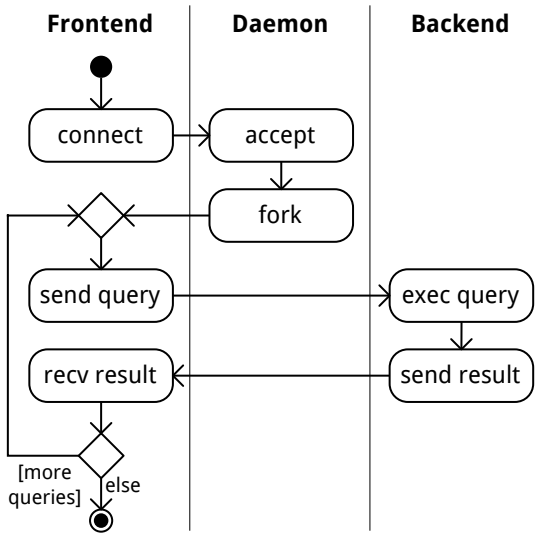

Рис. 2. Взаимодействие клиента и сервера

Порядок взаимодействия клиента и СУБД представлен на рис. 2. Сначала клиент устанавливает соединение с демоном. Демон принимает соединение от клиента и затем с помощью системного вызова `fork()` создает серверный процесс. После этого клиент отправляет запрос серверному процессу, который выполняет обработку этого запроса и отправку результатов обратно клиенту.

Обработка запроса состоит из следующих этапов:

- *parse* — разбор запроса на языке SQL;
- *rewrite* — преобразование запроса;
- *plan/optimize* — составление плана запроса и его оптимизация;

- *execute* — выполнение плана запроса.

СУБД PostgreSQL содержит следующие подсистемы, представленные на рис. 3:

- *Parser* — подсистема, которая осуществляет разбор SQL-запросов;
- *Rewriter* — подсистема, выполняющая преобразование запроса в соответствии с правилами подстановки, которые хранятся в базе данных (например, для реализации представлений);
- *Storage* — подсистема хранения данных и метаданных;
- *Planner* — подсистема, которая выполняет составление плана запроса;
- *Executor* — подсистема, исполняющая план запроса;
- *libpq* — библиотека, реализующая протокол взаимодействия клиента (libpq-fe) и сервера (libpq-be).

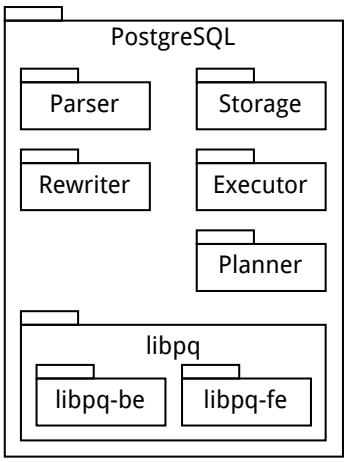

Рис. 3. Подсистемы СУБД PostgreSQL

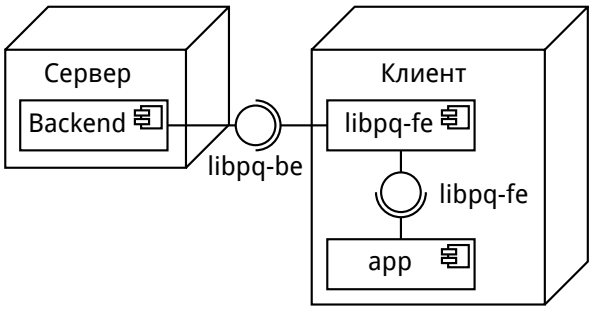

Рис. 4. Размещение компонентов

Размещение компонентов СУБД PostgreSQL приведено на рис. 4.

На клиенте размещается библиотека `libpq` и приложение пользователя. Все остальные компоненты размещаются на узлах сервера.

3. Архитектура PargreSQL

PargreSQL использует идею фрагментного параллелизма [3]. Общая схема параллельной обработки запроса представлена на рис. 5. Каждое отношение (таблица) базы данных делится на горизонтальные *фрагменты*, распределяемые по процессорным узлам вычислительной системы. Способ фрагментации определяется *функцией фрагментации*, вычисляющей для каждого кортежа отношения номер процессорного узла, на котором должен быть размещен этот кортеж. Запрос выполняется в виде нескольких параллельных процессов (*агентов*), каждый из которых обрабатывает отдельный фрагмент отношения. Полученные фрагменты сливаются в результирующее отношение.

Архитектура клиент-серверного взаимодействия параллельной СУБД PargreSQL, в отличие от последовательной СУБД PostgreSQL, предполагает, что клиент взаимодействует с двумя или более серверами одновременно (см. рис. 6).

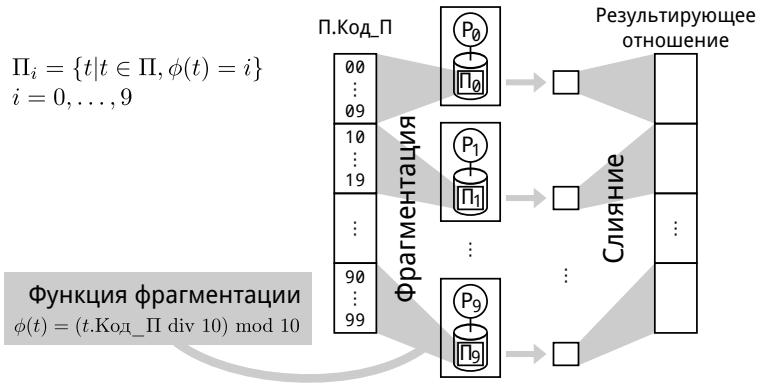

Рис. 5. Параллельная обработка запроса на основе фрагментного параллелизма

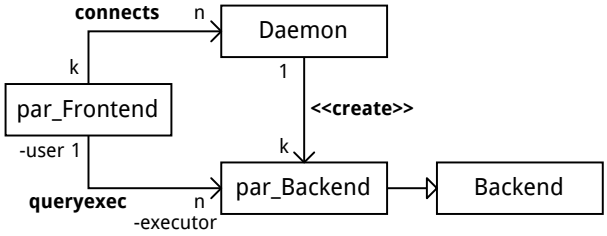

Рис. 6. Процессы СУБД PargreSQL

Порядок взаимодействия клиента и СУБД PargreSQL представлен на рис. 7. Клиентское приложение подключается сразу ко всем экземплярам СУБД и отправляет им одинаковый запрос.

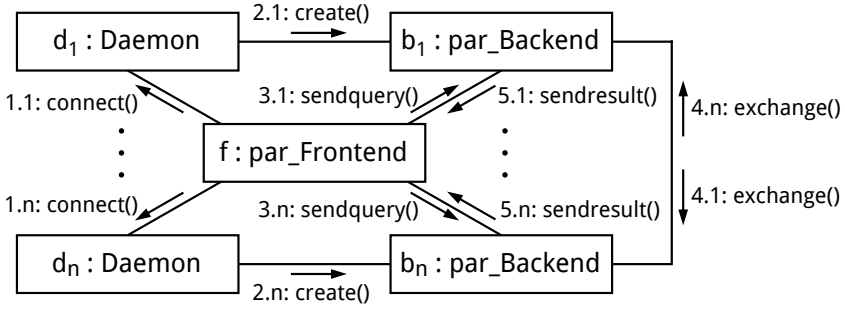

Рис. 7. Взаимодействие клиента и серверов PargreSQL

Параллельная обработка запроса состоит из следующих этапов:

- *parse* — разбор запроса на языке SQL;
- *rewrite* — преобразование запроса;
- *plan/optimize* — составление последовательного плана запроса и его оптимизация;
- *parallelize* — формирование параллельного плана запроса на основе последовательного путем вставки операторов *exchange*;
- *execute* — выполнение плана запроса;
- *balance* — балансировка загрузки серверных процессов.

Архитектура PargreSQL представлена на рис. 8. PostgreSQL является подсистемой в рамках системы PargreSQL. Для разработки PargreSQL необходимо внести изменения в исходные тексты следующих подсистем PostgreSQL: Storage, Executor и Planner.

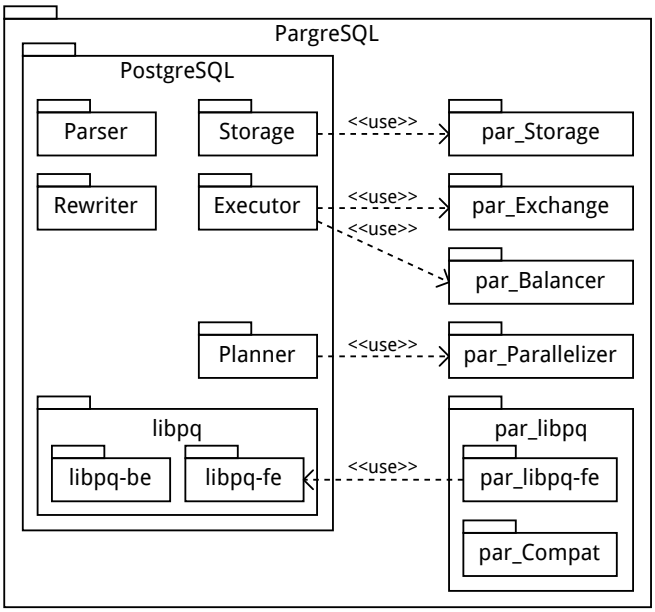

Рис. 8. Архитектура СУБД PargreSQL

Данные изменения обеспечивают внедрение следующих новых подсистем:

- *par_Storage* — подсистема хранения данных о фрагментации отношений;
- *par_Exchange* — подсистема, реализующая оператор *exchange* [3], который выполняет обмен кортежами между экземплярами СУБД;
- *par_Parallelizer* — подсистема, выполняющая добавление в нужные места последовательного плана запроса операторов *exhange*;
- *par_Balancer* — подсистема, выполняющая динамическую балансировку загрузки серверных процессов.

Задача оператора *exchange* — передать все кортежи, которые должны быть обработаны ядрами PargreSQL на других вычислительных узлах, и получить все кортежи, предназначенные для обработки ядром PargreSQL на данном 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 дерева запроса.

В PargreSQL также входят следующие новые подсистемы, которые не требуют изменения оригинальных подсистем PostgreSQL:

- *par_libpq-fe* — надстройка над `libpq-fe`, реализующая тиражирование запроса;
- *par_Compat* — подсистема, реализующая прозрачное для приложения подключение `par_libpq-fe`.

Размещение компонентов СУБД PargreSQL приведено на рис. 9.

На клиенте размещаются библиотеки `par_libpq` и `libpq-fe` и приложение пользователя вместе с конфигурационным файлом в формате XML, в котором определяются параметры работы приложения (адреса узлов, фрагментация таблиц и др.). Остальные компоненты размещаются на узлах сервера.

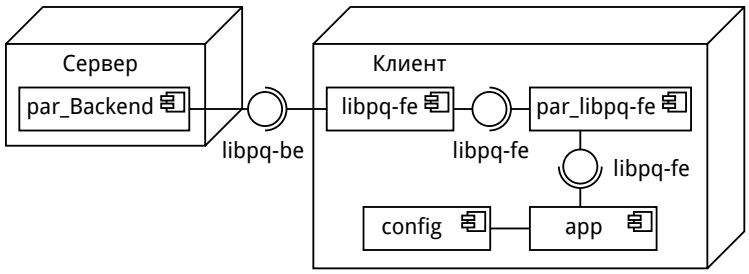

Рис. 9. Размещение компонентов PargreSQL

4. Принципы реализации PargreSQL

PargreSQL разрабатывается в соответствии со следующими основными принципами: масштабируемость, минимальность и прозрачность.

Масштабируемость заключается в том, что параллельная СУБД PargreSQL, запущенная на одном вычислительном узле, работает так же, как последовательная СУБД PostgreSQL.

Масштабируемость реализуется путем использования оператора *exchange* [3]. Оператор *exchange* вычисляет значение функции пересылки для каждого поступающего кортежа и передает кортеж на вычислительный узел, номер которого совпадает со значением функции пересылки.

Таким образом, оператор *exchange* не изменяет кортеж (передает его в вышележащий узел плана), если значение функции пересылки совпадает с номером текущего узла. Когда PargreSQL запускается на одном узле, функция пересылки тождественно равна номеру единственного узла — нулю.

В исходные тексты PostgreSQL вносятся *минимальные* изменения. Изменения в структурах данных и алгоритмах инкапсулируются в новых файлах исходных текстов, подключаемых к исходным текстам PostgreSQL.

```
// someheader.h

typedef struct
{
    char field1;
    char field2;
} somestruct;
```

⇝

```
// someheader.h

typedef struct
{
    char field1;
    char field2;
    par_somestruct newfields;
} somestruct;
```

+

```
// par_someheader.h

typedef struct
{
    char newfield1;
    char newfield2;
} par_somestruct;
```

Рис. 10. Техника внесения изменений в определения структур данных

На рис. 10 показан пример применения данного подхода для добавления новых полей в оригинальную структуру данных. В новом файле описывается тип `par_somestruct`, содержащий новые поля, а в оригинильную структуру добавляется новое поле типа `par_somestruct`.

На рис. 11 показан пример применения данного подхода для изменения оригинальных алгоритмов. В тело оригинальной функции добавляется вызов новой функции `newfunc()`, а сама функция `newfunc()` определяется в файле исходных текстов новой подсистемы.

Использование PargreSQL является *прозрачным* для пользовательских приложений. Подключение PargreSQL к прикладным программам, которые до этого использовали PostgreSQL, производится с минимальными изменениями в исходных кодах приложения.

Прозрачность реализуется следующим образом. Пользовательское приложение вместе с заголовочным файлом `par_libpq-fe.h` подключает заголовочный файл `par_Compat.h`. Этот

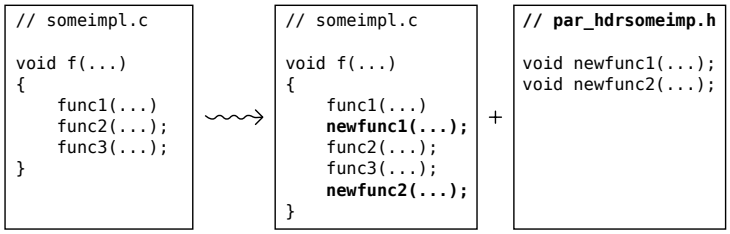

Рис. 11. Техника внесения изменений в исходные тексты функций

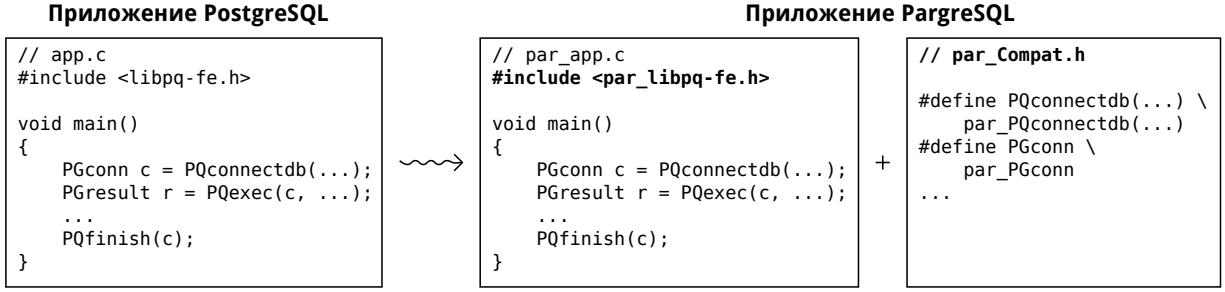

Рис. 12. Прозрачность использования `par_libpq`

файл содержит объявление макросов, заменяющих вызовы функций подсистемы `libpq` на вызовы функций подсистемы `par_libpq`. Таким образом, для адаптации PostgreSQL-приложения в исходном тексте приложения требуется изменение одной строки кода.

На рис. 12 показан прозрачный способ подключения `par_libpq`.

5. Заключение

В данной работе описана архитектура и принципы реализации параллельной СУБД PargreSQL для многопроцессорных вычислительных систем с кластерной архитектурой. PargreSQL основана на свободной СУБД PostgreSQL и использует фрагментный параллелизм.

Направлением дальнейших исследований является завершение разработки PargreSQL и проведение экспериментов по исследованию ее ускорения и масштабируемости.

Работа выполнена при финансовой поддержке Минобрнауки РФ (государственный контракт №07.514.11.4036) и Российского фонда фундаментальных исследований (проект 12-07-00443-а).

Статья рекомендована к публикации программным комитетом международной научной конференции «Параллельные вычислительные технологии 2011».

Литература

1. Stonebraker, M. The POSTGRES next-generation database management system / M. Stonebraker, G. Kemnitz // Communications of the ACM. – Oct. 1991. – V. 34, № 10. – P. 78 – 92.

2. Sokolinsky, L. Omega: The Highly Parallel Database System Project / L. Sokolinsky, O. Axenov, S. Gutova // Proceedings of the First East-European Symposium on Advances in Database and Information Systems (ADBIS'97), St.-Petersburg, September 2 – 5, 1997. – St.-Petersburg: Nevsky Dialect, 1997. – V. 2. – P. 88 – 90.

3. Соколинский, Л.Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой / Л.Б. Соколинский // Программирование. – 2001. – № 6. – С. 13 – 29.

4. Samokhvalov, N. XML Support in PostgreSQL / N. Samokhvalov // SYRCoDIS, CEUR Workshop Proceedings. – 2007. – V. 256. – P. 1 – 6.

5. Havinga, Y. Adding HL7 version 3 data types to PostgreSQL / Y. Havinga, W. Dijkstra, A. de Keijzer // Computing Research Repository. – 2010. – abs/1003.3370.

6. POSTGRESQL-IE: An Image-handling Extension for PostgreSQL / D. Guliato, E.V. de Melo, R.M. Rangayyan, R.C. Soares // J. of Digital Imaging. – 2009 – V. 22, № 2. – P. 149 – 165.

7. Levshin, D.V. Algorithms for integrating PostgreSQL with the semantic web / D.V. Levshin, A.S. Markov // Programming and Computer Software. – 2009. – V. 35, № 3. – P. 136 – 144.

8. Lee, R. Extending PostgreSQL to Support Distributed/Heterogeneous Query Processing / R. Lee, M. Zhou // Database Systems for Advanced Applications. Lecture Notes in Computer Science. – Springer, 2007. – V. 4443. – P. 1086 – 1097.

9. High-Performance Query Processing of a Real-World OLAP Database with ParGRES / M. Paes, A.A.B. Lima, P. Valduriez, M. Mattoso // VECPAR, Lecture Notes in Computer Science. – Springer, 2008. – V. 5336. – P. 188 – 200.

10. Kotowski, N. Parallel query processing for OLAP in grids / N. Kotowski, A.A.B Lima, E. Pacitti, P. Valduriez, M. Mattoso // Concurrency and Computation: Practice and Experience. – 2008. – V. 20, № 17. – P. 2039 – 2048.

11. DeWitt, D.J. Parallel Database Systems: The Future of High Performance Database Systems / D.J. DeWitt, J. Gray // Communications of the ACM. – 1992. – V. 35, № 6. – P. 85 – 98.

12. Sokolinsky, L.B. Organization of Parallel Query Processing in Multiprocessor Database Machines with Hierarchical Architecture / L.B. Sokolinsky // Programming and Computer Software. – 2001. – V. 27, № 6. – P. 297 – 308.

13. Lepikhov, A.V. Query processing in a DBMS for cluster systems / A.V. Lepikhov, L.B. Sokolinsky // Programming and Computer Software. – 2010. – V. 36, № 4. – P. 205 – 215.

Михаил Леонидович Цымблер, кандидат физико-математических наук, доцент, кафедра системного программирования, Южно-Уральский государственный университет (г. Челябинск, Российская Федерация), zymbler@gmail.com

Константин Сергеевич Пан, кафедра системного программирования, Южно-Уральский государственный университет (г. Челябинск, Российская Федерация), kvapen@gmail.com

---

MSC 68P15

Development of a parallel Database Management System on the Basis of Open-Source POSTGRESQL DBMS

C.S. Pan, South Ural State University (Chelyabinsk, Russian Federation)**,**
M.L. Zymbler, South Ural State University (Chelyabinsk, Russian Federation)

The paper describes the architecture and implementation of PargreSQL parallel database management system (DBMS) for distributed memory multiprocessors. PargreSQL is based upon PostgreSQL open-source DBMS and exploits partitioned parallelism. The paper is devoted to development of a parallel database management system (DBMS) by means of embedding of the parallel query execution techniques based on partitioning parallelism concept into open-source PostgreSQL DBMS. The architecture and implementation principles of the parallel DBMS for cluster computing systems are described. PostgreSQL is a subsystem of PargreSQL. The necessary modifications of the PostgreSQL subsystems are described. These modifications suppose minimal changes in the source code. The changes in data structures and algorithms are encapsulated into separate source code files that are included into the original project file structure. The usage of PargreSQL is transparent for applications. It demands minimal modifications of an application's source code. PargreSQL running on one computing node works like PostgreSQL.

Keywords: parallel DBMS, partitioned parallelism, PostgreSQL.

References

1. Stonebraker M., Kemnitz G. The POSTGRES Next-generation Database Danagement System. *Communications of the ACM.* Oct. 1991, vol. 34, no. 10, pp. 78 – 92.
2. Sokolinsky L., Axenov O., Gutova S. Omega: The Highly Parallel Database System Project. *Proceedings of the First East-European Symposium on Advances in Database and Information Systems (ADBIS'97), St.-Petersburg, September 2 – 5*, 1997, vol. 2, pp. 88 – 90.
3. Sokolinsky L.B. Organization of Parallel Query Processing in Multiprocessor Database Machines with Hierarchical Architecture. *Programming and Computer Software*, 2001, vol. 27, no. 6, pp. 297 – 308.
4. Samokhvalov N. XML Support in PostgreSQL. *SYRCoDIS, CEUR Workshop Proceedings*, 2007, vol. 256, pp. 1 – 6.
5. Havinga Y., Dijkstra W., de Keijzer A. Adding HL7 Version 3 Data Types to PostgreSQL. *Computing Research Repository*, 2010, vol. abs/1003.3370.
6. Guliato D., de Melo E.V., Rangayyan R.M., Soares R.C. POSTGRESQL-IE: An Image-handling Extension for PostgreSQL. *Journal of Digital Imaging*, 2009, vol. 22, no. 2, pp. 149 – 165.
7. Levshin D.V., Markov A.S. Algorithms for Integrating PostgreSQL with the Semantic Web. *Programming and Computer Software*, 2009, vol. 35, no. 3, pp. 136 – 144.
8. Lee R., Zhou M. Extending PostgreSQL to Support Distributed/Heterogeneous Query Processing. *Database Systems for Advanced Applications. Lecture Notes in Computer Science.* 2007, vol. 4443, pp. 1086 – 1097.
9. Paes M., Lima A.A.B., Valduriez P., Mattoso M. High-Performance Query Processing of a Real-World OLAP Database with ParGRES. *VECPAR, Lecture Notes in Computer Science*, 2008, vol. 5336, pp. 188–200.
10. Kotowski N., Lima A.A.B, Pacitti E., Valduriez P., Mattoso M. Parallel Query Processing for OLAP in Grids. *Concurrency and Computation: Practice and Experience*, 2008, vol. 20, no. 17, pp. 2039 – 2048.
11. DeWitt D.J., Gray J. Parallel Database Systems: The Future of High Performance Database Systems. *Communications of the ACM*, 1992, vol. 35, no. 6, pp. 85 – 98.
12. Sokolinsky L.B. Organization of Parallel Query Processing in Multiprocessor Database Machines with Hierarchical Architecture. *Programming and Computer Software*, 2001, vol. 27, no. 6, pp. 297 – 308.
13. Lepikhov A.V., Sokolinsky L.B. Query Processing in a DBMS for Cluster Systems. *Programming and Computer Software*, 2010, vol. 36, no. 4, pp. 205 – 215.

Поступила в редакцию 26 июля 2011 г.