УДК 004.65

ПАРАЛЛЕЛЬНАЯ СУБД С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ ДЛЯ КЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Е.В. Гавриш, А.В. Колтаков, А.А. Медведев, Л.Б. Соколинский

Статья посвящена вопросам разработки параллельной СУБД с открытым исходным кодом для кластерных вычислительных систем. Дан обзор известных решений в этой области. Рассмотрена новая параллельная СУБД «Омега» с открытым исходным кодом, ориентированная на кластерные вычислительные системы. Приведена общая архитектура системы «Омега». Представлены диаграмма размещения и диаграмма классов. Описаны основные подсистемы СУБД «Омега» и принципы их взаимодействия при выполнении запросов.

Ключевые слова: параллельная СУБД, обработка сверхбольших баз данных, программное обеспечение с открытым исходным кодом, кластерные вычислительные системы.

Введение

Проблема больших данных становится все более актуальной в наше время. К 2020 г. объем цифровой информации, хранимой в базах данных, достигнет отметки в 40 зеттабайт [13]. Наиболее эффективным решением проблемы хранения и обработки больших баз данных является использование параллельных систем управления базами данных (СУБД), обеспечивающих параллельную обработку запросов на многопроцессорных вычислительных системах. Среди многопроцессорных вычислительных систем сегодня выделяются кластеры, занимающие в списке Top500 [16] более 80 %. Кластеры обладают хорошим соотношением цена/производительность. Это объясняется тем, что кластеры состоят из компонент, которые массово продаются на рынке.

Известно несколько коммерческих решений, которые позволяют обрабатывать большие объемы данных, но они до последнего времени были ориентированы на ту или иную специфическую аппаратную платформу (DB2 Parallel Edition, NonStop SQL, NCR Teradata, Oracle RAC, Greenplum и др.) и не подходили для массового использования на кластерных системах. Кроме того, указанные коммерческие решения являются дорогостоящими. Так, аппаратно-программные решения компании Teradata предлагаются по ценам от $41 000 за 1 терабайт обрабатываемой базы данных [15]. Сегодня появляются на рынке коммерческие СУБД, ориентированные на кластерные системы. Однако эти продукты также отличаются высокой стоимостью. Например, компания Oracle предлагает СУБД Oracle RAC, предназначенную для обработки запросов на кластерных системах, по цене $10 000 за процессор при бессрочной лицензии [9].

С другой стороны, набирают популярность расширения для свободно распространяемых СУБД с открытым исходным кодом, которые обеспечивают параллельную обработку транзакций [7, 11, 20].

Данная работа посвящена вопросам разработки свободно распространяемой параллельной СУБД для кластерных систем, которая должна обладать высокой масштабируемостью, однако может проигрывать коммерческим аналогам в производительности. В статье описывается подобная параллельная СУБД, основанная на реляционной модели данных. Раздел 1 посвящен обзору известных СУБД с открытым исходным кодом,

ориентированных на обработку сверхбольших баз данных. В разделе 2 представлена архитектура разрабатываемой параллельной СУБД и описана общая схема обработки запросов. В разделе 3 представлены диаграммы классов основных подсистем СУБД «Омега». Раздел 4 содержит описание протокола взаимодействия пользовательского приложения и СУБД «Омега». В заключении представлены основные полученные результаты и направления дальнейших исследований.

Обзор известных решений

Рассмотрим известные параллельные СУБД с открытым исходным кодом, ориентированные на хранение и обработку сверхбольших баз данных.

СУБД SciDB [3] ориентирована на обработку научных данных, полученных в результате экспериментов и наблюдений. Данная СУБД оптимизирована для обработки и анализа «сырых» данных, которые интенсивно читаются, но почти не изменяются. SciDB не рассчитана на обработку транзакций в реальном времени (OLTP), не поддерживает ACID-транзакции и не является реляционной, так как хранение данных организовано в виде многомерных вложенных массивов, для обработки которых вместо языка SQL задействованы языки AQL (Array Query Language) и AFL (Array Functional Language).

Система HadoopDB [1] основана на концепции связывания нескольких одноузловых СУБД (например, MySQL или PostgreSQL) с помощью технологий Hadoop в единую параллельную СУБД. Среда Hadoop имеет два уровня системной иерархии: уровень хранения данных и уровень обработки данных. Уровень хранения данных представлен распределенной файловой системой HDFS (Hadoop Distributed File System), а уровень обработки данных реализуется с помощью программного каркаса Hadoop MapReduce. Основным недостатком HadoopDB является низкая производительность по сравнению с реляционными параллельными СУБД [12, 18]. Это объясняется тем, что изначально система HadoopDB разрабатывалась как прототип параллельной системы управления аналитическими (научными) данными.

СУБД MongoDB [2] является документо-ориентированной СУБД класса NoSQL. В MongoDB отсутствует полноценная поддержка ACID-транзакций. Существенным недостатком данной СУБД является полное отсутствие поддержки изолированности операций над данными.

Свободная параллельная СУБД MySQL Cluster [8, 14] обладает хорошей производительностью и масштабируемостью на простых запросах, но эффективность СУБД существенно падает при обработке сложных запросов с условиями [4]. Недостатком MySQL Cluster является использование полной репликации данных.

В рамках научного проекта ParGRES [10] разрабатывается параллельная СУБД, предназначенная для обработки OLAP-запросов. СУБД ParGRES представляет собой промежуточное программное обеспечение, которое управляет экземплярами свободной последовательной СУБД PostgreSQL, запускаемыми на узлах кластерной системы. Недостатком СУБД ParGRES является использование полной репликации всех таблиц базы данных на узлах кластерной системы, что приводит к общему снижению производительности СУБД. Развитием данной разработки является СУБД GParGRES [6], предназначенная для грид-сред. СУБД GParGRES использует репликацию базы данных, межзапросный и внутризапросный параллелизм для эффективной обработки OLAP-

запросов в грид. Предложенный в СУБД GParGRES подход подразумевает распараллеливание запроса на двух уровнях: на уровне грид (реализовано в GParGRES) и на уровне узлов (реализовано в ParGRES). Однако СУБД GParGRES также требует полной репликации базы данных на всех узлах всех кластеров, объединенных в грид.

Параллельная СУБД VoltDB [17] с открытым исходным кодом предназначена для кластерных систем и обладает сравнительно высокой производительностью. СУБД является представителем класса NoSQL. Существенным недостатком данной СУБД с точки зрения обработки сверхбольших баз данных является необходимость хранить всю базу данных в оперативной памяти.

Проведенный анализ показывает, что на данный момент отсутствуют реляционные параллельные СУБД с открытым исходным кодом, ориентированные на кластерную архитектуру и обладающие высокой эффективностью при обработке запросов.

Общая архитектура СУБД «Омега»

Целью проекта «Омега», выполняемого в Лаборатории суперкомпьютерного моделирования Южно-Уральского государственного университета, является разработка новых методов и алгоритмов параллельной обработки запросов к базам данных, ориентированных на современные кластерные вычислительные системы, а также реализация этих методов и алгоритмов в виде параллельной СУБД «Омега» с открытым исходным кодом. Фундаментом разрабатываемой параллельной СУБД с одноименным названием «Омега» является архитектура, представленная на рис. 1.

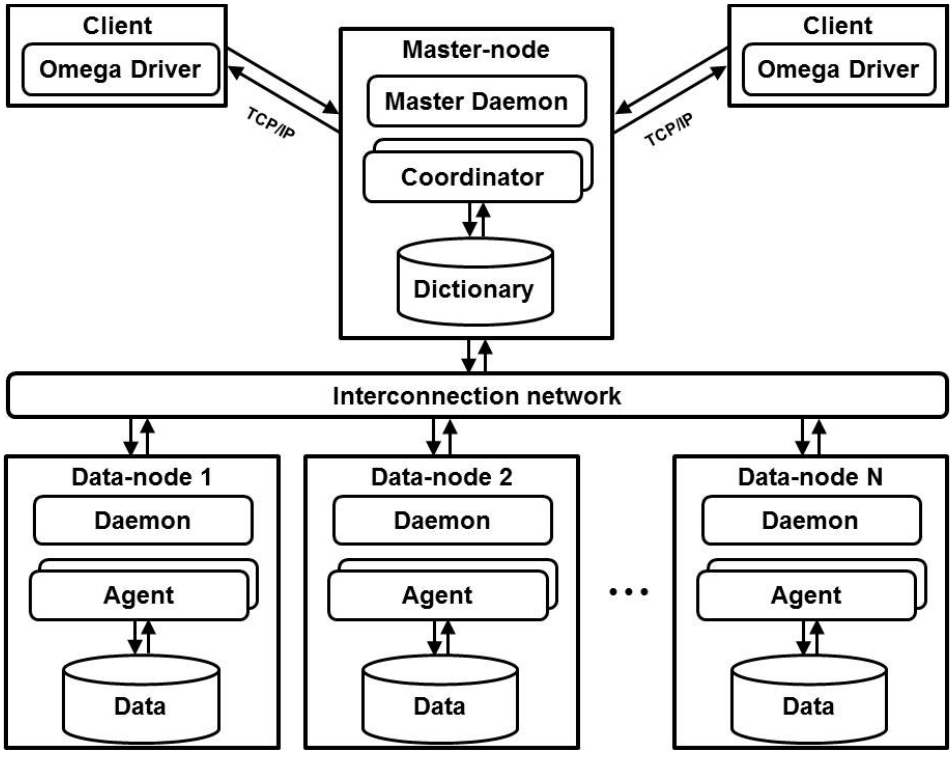

Рис. 1. Общая архитектура СУБД «Омега»

В соответствии с данной архитектурой в СУБД выделяются следующие основные подсистемы.

Управляющий узел (Master-node) – это совокупность процессов и подсистем, которые отвечают за взаимодействие СУБД с клиентами, скрывая от них все детали распределенной обработки запросов. На данном узле размещается *словарь базы данных* (*Dictionary*). На управляющем узле выполняется *серверный фоновый процесс* (*MasterDaemon*), который отвечает за прием соединений, устанавливаемых клиентами. Для каждого клиента серверный фоновый процесс порождает специальный процесс – *координатор* (*Coordinator*) для обработки запросов. Координатор выполняет построение плана запроса и его рассылку на вычислительные узлы, сбор промежуточных результатов выполнения запроса и отправку конечного результата клиенту. Данный процесс также управляет балансировкой загрузки СУБД во время обработки запроса.

Поскольку для каждого клиента создается отдельный координатор, для однозначной идентификации координатора внутри системы вводится понятие сессии. *Сессия* – это совокупность процессов СУБД, выполняемых одним клиентом. Каждому отдельному клиенту присваивается уникальный номер сессии.

Вычислительный узел (Data-node) – это совокупность процессов и подсистем, которые непосредственно отвечают за исполнение запроса и хранение *данных* (*Data*). Количество таких узлов теоретически ограничено производственными мощностями вычислительной системы. На данном узле выполняется *фоновый процесс* (*Daemon*), который взаимодействует с серверным фоновым процессом (MasterDaemon) и при получении соответствующей команды создает отдельный процесс – параллельный *агент* (*Agent*) [19] для обработки запроса. Агент получает от координатора план запроса, исполняет его и отсылает координатору промежуточный результат. Агенты однозначно сопоставляются с координатором по номеру сессии.

Клиент (Client) – это пользовательское (клиентское) приложение на языке Java, которое для работы с СУБД использует библиотеку *Omega Driver*. *Omega Driver* представляет собой JDBC-драйвер [5], который обеспечивает доступ к СУБД по протоколу TCP/IP через стандартные API-интерфейсы JDBC, доступные на платформе Java.

Взаимодействие между узлами происходит с использованием высокоскоростной вычислительной сети – *interconnection network*.

Рассмотрим порядок взаимодействия основных модулей СУБД при выполнении запроса (рис. 2). Порядок взаимодействия включает в себя следующие шаги.

1. Connect (соединение). Клиент, используя Omega Driver, устанавливает соединение с серверным процессом.
2. Accept (прием). Серверный процесс принимает запрос клиента на соединение и связывает себя с конкретным адресом, для того, чтобы иметь возможность в дальнейшем обрабатывать запросы клиента.
3. CreateCoordinator (создание координатора). Серверный процесс создает отдельный процесс-координатор для данного клиента. На данном шаге координатору присваивается номер сессии.
4. SendSession (отправка сессии). Серверный процесс отправляет фоновым процессам на вычислительных узлах номер сессии данного клиента.
5. CreateAgent (создание агента). Каждый фоновый процесс создает параллельного агента и присваивает ему номер сессии.
6. Execute (выполнение запроса). Клиент отсылает координатору запрос на языке SQL.

7. SendMetadata (отправка метаданных). Координатор обрабатывает запрос, строит план выполнения запроса, вычисляет метаданные результирующего отношения и отправляет их клиенту.
8. SendPlan (отправка плана). Координатор производит пересылку плана выполнения запроса его агентам.
9. Exchange (обмен). Агенты исполняют план запроса. Во время исполнения плана агенты могут осуществлять межпроцессорные обмены (пересылку кортежей по сети).
10. SendTuple (отправка кортежа). Агенты отсылают результирующие кортежи координатору.
11. SendResult (отправка результата). Координатор производит слияние и обработку кортежей, полученных от агентов, и отсылает конечный результат клиенту.

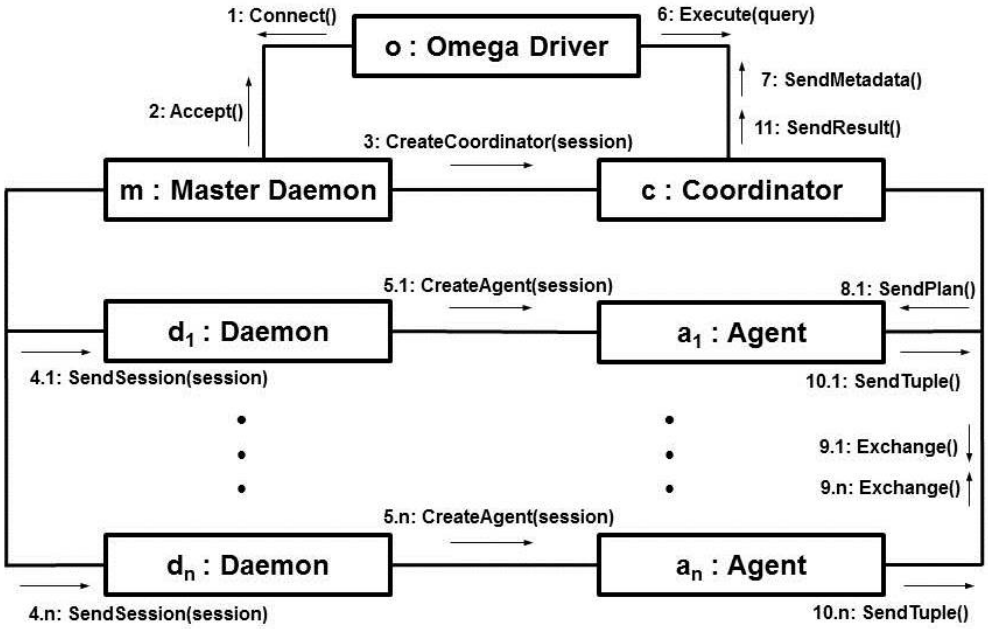

Рис. 2. Схема обработки запроса

Структура СУБД «Омега»

Ключевым элементом СУБД «Омега» является управляющий узел. Управляющий узел включает в себя 4 подсистемы: серверный фоновый процесс, координатор, менеджер сессий и словарь баз данных. Соответствующая диаграмма классов представлена на рис. 3.

Серверный фоновый процесс включает в себя следующие основные методы:

- run – запускает основной цикл работы серверного процесса;
- accept – принимает запрос на соединение от клиента;
- createCoordinator – создает процесс-координатор и предает ему номер сессии;
- sendSession – передает фоновым процессам на вычислительных узлах номер сессии;
- finish – вызывается при поступлении команды завершения СУБД, корректно завершает все запущенные процессы.

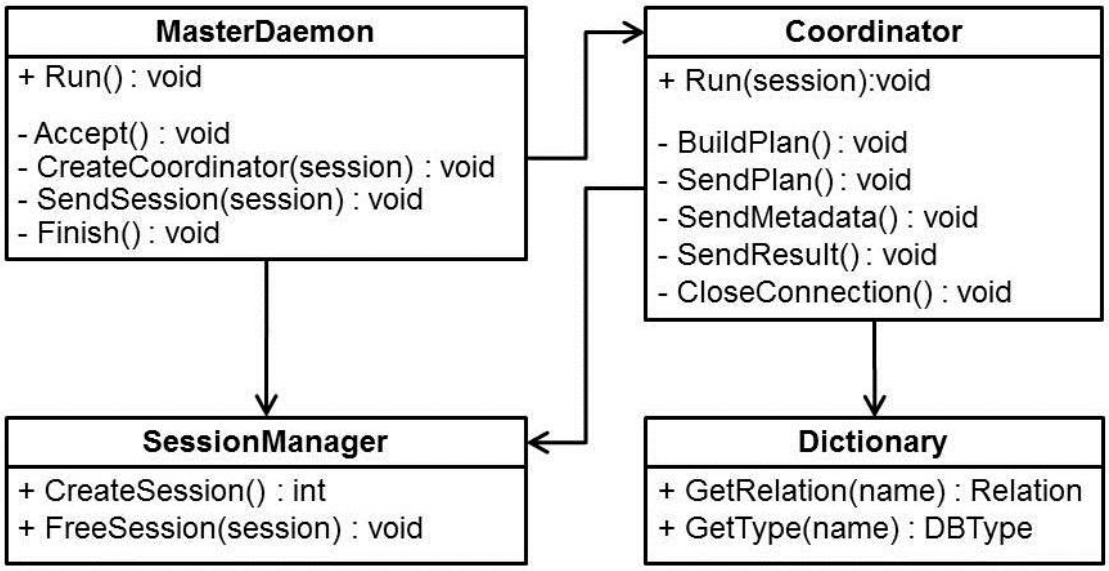

Рис. 3. Диаграмма классов управляющего узла

Координатор включает в себя следующие основные методы:

- run – запускает основной цикл работы серверного процесса;
- buildPlan – производит обработку запроса на языке SQL и выполняет построение плана запроса;
- sendPlan – отправляет план запроса соответствующим параллельным агентам;
- sendMetadata – отправляет клиенту метаданные результирующего отношения;
- sendResult – отправляет клиенту результат выполнения запроса;
- closeConnection – разрывает соединение с клиентом, освобождает сессию.

Менеджер сессий включает в себя следующие основные методы:

- createSession – осуществляет создание новой сессии, возвращает ее номер;
- freeSession – осуществляет освобождение сессии;

Словарь баз данных включает в себя следующие основные методы:

- getRelation – возвращает информацию об отношении;
- getType – возвращает информацию о типе.

Взаимодействие серверного процесса СУБД и клиентского приложения осуществляется с помощью стандартных API-интерфейсов JDBC, доступных на платформе Java. В рамках проекта реализовано подмножество функций стандарта JDBC. Диаграмма классов клиента представлена на рис. 4.

В классах DriverManager и Driver предусмотрены методы getConnection и connect соответственно для установки соединения между клиентом и управляющим узлом СУБД. Параметром данных методов является символьная строка, которая имеет формат «jdbc:<тип драйвера>://<ip-адрес сервера>:<порт>».

Основные методы класса Connection:

- createStatement – возвращает объекты типа Statement, служащие для исполнения запросов к базе данных на языке SQL;
- close – осуществляет закрытие соединения.

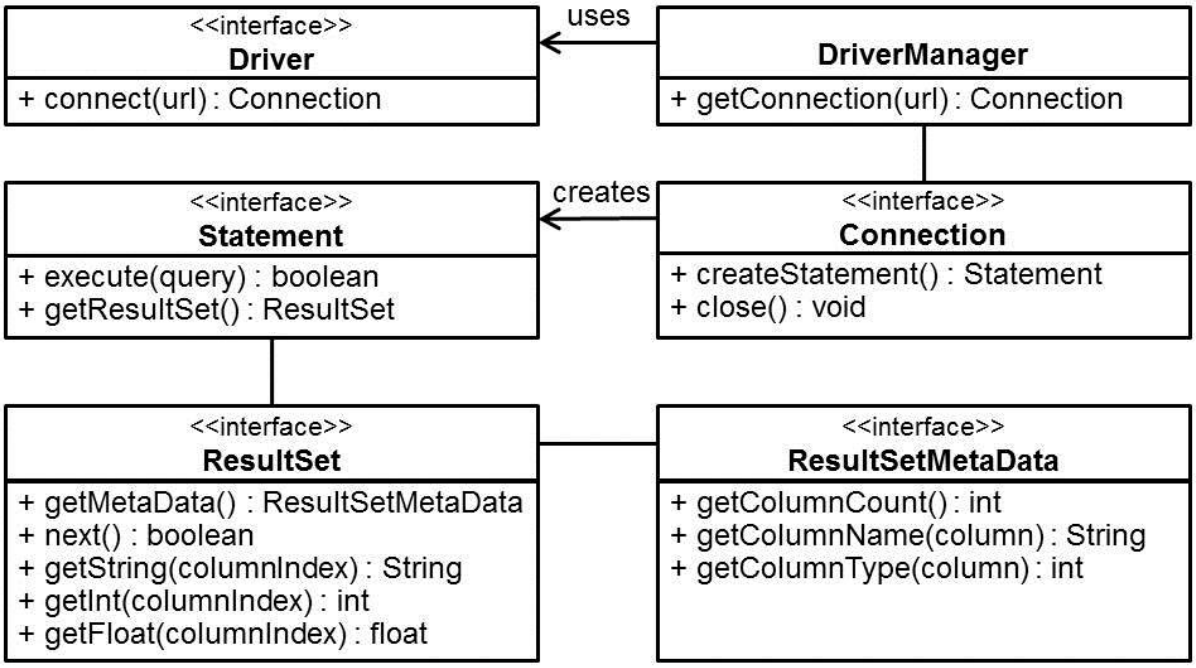

Рис. 4. Диаграмма классов клиента

Основные методы класса Statement:

- execute – возвращает истину, если запрос является запросом типа SELECT (существует ResultSet), иначе возвращает ложь, то есть имеет место быть запрос типа INSERT или UPDATE (отсутствует ResultSet);
- getResultSet – возвращает множество строк, удовлетворяющих выражению на языке SQL.

Основные методы класса ResultSet:

- getMetadata – возвращает метаданные;
- next – возвращает истину, если существует следующий кортеж, иначе возвращает ложь;
- getString, getInt и getFloat – возвращают значение конкретного поля текущего кортежа.

Основные методы класса ResultSetMetaData:

- getColumnCount – возвращает количество столбцов (колонок);
- getColumnName – возвращает название поля столбца (колонки);
- getColumnType – возвращает идентификатор типа данных столбца (колонки), где для типа int соответствует значение 4, float – 6, а string – 15.

Протокол взаимодействия клиента и СУБД

Взаимодействие клиента (пользовательского приложения) и СУБД (управляющего узла) осуществляется с помощью протокола на основе специальных команд. Протокол определяет формат команд и их последовательность при обработке запросов от клиента.

Формат команды следующий: <идентификатор><размер данных><данные>.

Идентификатор команды определяет семантику команды, наличие передаваемых данных и их назначение. Расшифровка идентификаторов, используемых в протоколе, представлена в таблице 1.

Таблица 1.

Идентификаторы протокола

Идентификаторы клиента			
№	id	Команда	Семантика
1	S	start	Установить соединение.
2	Q	query	Отправить запрос.
3	C	close	Закрыть соединение.
Идентификаторы управляющего узла			
№	id	Команда	Семантика
1	R	ready	Готовность к обработке запросов.
2	I	info	Отправить информацию о выполнении запроса.
3	M	metadata	Отправить метаданные.
4	T	tuple	Отправить кортеж.
5	E	error	Уведомить об ошибке.
6	F	finish	Завершить обработку запроса.

Заключение

В работе рассмотрены основные современные свободные решения в сфере обработки сверхбольших баз данных, представлена архитектура разрабатываемой авторами СУБД, приведены диаграммы классов основных подсистем и описан процесс выполнения запросов в рамках данной архитектуры.

На данный момент авторами работы разработан прототип, который успешно развертывается на узлах вычислительного кластера и осуществляет взаимодействие между узлами СУБД (основные системные команды).

В рамках продолжения исследований планируется реализация основных компонентов параллельной СУБД с использованием методов и алгоритмов параллельной обработки запросов в СУБД.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 12-07-00443-а.

Литература

1. Abouzeid, A. HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads / A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, A. Rasin // VLDB'2009, Proceedings of 35th International Conference on Very Large Data Bases, August 24-28, 2009, Lyon, France. – VLDB Endowment, 2009. – P. 922–933.
2. Boicea, A. MongoDB vs Oracle – database comparison / A. Boicea, F. Radulescu, L.I. Agapin // Proceedings of the Third International Conference on Emerging Intelligent Data and Web Technologies, September 19-21, 2012, Bucharst, Romania. – P. 330–335.
3. Brown, P.G. Overview of sciDB: Large Scale Array Storage, processing and analysis / P.G. Brown // Proceedings of the ACM SIGMOD International Conference on Man-

agement of Data, June 6-10, 2010, Indianapolis, Indiana, USA. – ACM, 2010. – P. 963–968.

4. Hubel, M. Technical Comparison of DB2 and MySQL / M. Hubel – Martin Hubel Consulting Inc., 2004. – 32 p.
5. Java SE Documentation. URL: http://www.oracle.com/technetwork/java/javase/jdbc/ (дата обращения: 10.03.2013).
6. 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. – Vol. 20, No. 17. – P. 2039–2048.
7. Lee, R. Extending PostgreSQL to Support Distributed/Heterogeneous Query Processing / R. Lee, M. Zhou // Proceedings of the 12th International Conference on Database Systems for Advanced Applications, April 9-12, 2007, Bangkok, Thailand. – Proceedings. Lecture Notes in Computer Science, Springer, 2007. – Vol. 4443. – P. 1086–1097.
8. MySQL Cluster Information.
   URL: http://www.mysql.com/products/cluster/resources.html (дата обращения: 10.03.2013).
9. Oracle Store. URL: http://shop.oracle.com (дата обращения: 10.03.2013).
10. Paes, M. High-Performance Query Processing of a Real-World OLAP Database with ParGRES / M. Paes, A.A.B. Lima, P. Valduriez, M. Mattoso // High Performance Computing for Computational Science – VECPAR 2008: 8th International Conference, June 24-27, 2008, Toulouse, France. – Revised Selected Papers. Springer, 2008. – P. 188–200.
11. Paulson, L.D. Open Source Databases Move into the Marketplace / L.D. Paulson // Computer, – 2004. – Vol. 37, No. 7. – P. 13–15.
12. Pavlo, A. Comparison of Approaches to Large Scale Data Analysis / A. Pavlo, A. Rasin, S. Madden, M. Stonebraker, D. DeWitt, E. Paulson, L. Shrinivas, D.J. Abadi // Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, June 29 – July 2, 2009, Providence, Rhode Island, USA. – ACM, 2009. – P. 165–178.
13. Press Release EMC2. URL: http://www.emc.com/about/news/press/2012/20121211-01.htm (дата обращения: 10.03.2013).
14. Ronstrom, M. Recovery Principles in MySQL Cluster 5.1 / M. Ronstrom, J. Oreland // Proceedings of the 31st International Conference on Very Large Data Bases, August 30 – September 2, 2005, Trondheim, Norway. – ACM, 2005. – P. 1108–1115.
15. Teradata Purpose-Built Platform Pricing.
   URL: http://www.teradata.com/t/WorkArea/DownloadAsset.aspx?id=4682 (дата обращения: 10.03.2013).
16. Top500 List. URL: http://www.top500.org (дата обращения: 10.03.2013).
17. VoltDB Documentation. URL: http://voltdb.com/community/documentation (дата обращения: 10.03.2013).
18. Кузнецов, С.Д. MapReduce: внутри, снаружи или сбоку от параллельных СУБД? / С.Д. Кузнецов // Труды Института системного программирования РАН. – 2010. – Т. 19. – С. 35–70.
19. Лепихов, А.В. Обработка запросов в СУБД для кластерных систем / А.В. Лепихов, Л.Б. Соколинский // Программирование. – 2010. – № 4. – С. 25–39.

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

Гавриш Евгений Владимирович, аспирант кафедры системного программирования Южно-Уральского государственного университета (Челябинск, Российская Федерация), evgeniy.gavrish@gmail.com

Колтаков Алексей Владимирович, аспирант кафедры системного программирования Южно-Уральского государственного университета (Челябинск, Российская Федерация), aleksey.koltakov@gmail.com

Медведев Александр Андреевич, аспирант кафедры системного программирования Южно-Уральского государственного университета (Челябинск, Российская Федерация), medbedalex@gmail.com

Соколинский Леонид Борисович, доктор физ.-мат. наук, профессор кафедры системного программирования Южно-Уральского государственного университета (Челябинск, Российская Федерация), sokolinsky@acm.org

---

PARALLEL OPEN SOURCE DBMS FOR CLUSTER COMPUTING SYSTEMS

E.V. Gavrish, A.V. Koltakov, A.A. Medvedev, L.B. Sokolinsky

The article deals with problem of development open source parallel DBMS for cluster computing systems. The article describes overview of well-known solves in this area. It is considered a new parallel open source DMBS named Omega, which is oriented on cluster computing systems. The article shows the general architecture of system Omega, its deployment and class diagrams. The article describes Omega's main subsystems and principles of their interaction during query processing.

Keywords: parallel DBMS, big data processing, open source software, cluster computing systems.

References

1. Abouzeid A., Bajda-Pawlikowski K., Abadi D., Silberschatz A., Rasin A. HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. VLDB'2009, Proceedings of 35th International Conference on Very Large Data Bases, August 24-28, 2009, Lyon, France. – VLDB Endowment, 2009. P. 922–933.
2. Boicea A., Radulescu F., Agapin L.I. MongoDB vs Oracle – database comparison. Proceedings of the Third International Conference on Emerging Intelligent Data and Web Technologies, September 19-21, 2012, Bucharst, Romania. P. 330–335.
3. Brown P.G. Overview of sciDB: Large Scale Array Storage, processing and analysis. Proceedings of the ACM SIGMOD International Conference on Management of Data, June 6-10, 2010, Indianapolis, Indiana, USA. ACM, 2010. P. 963–968.
4. Hubel, M. Technical Comparison of DB2 and MySQL. Martin Hubel Consulting Inc., 2004. 32 p.

5. Java SE Documentation. URL: http://www.oracle.com/technetwork/java/javase/jdbc/ (accessed: 10.03.2013).
6. 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. P. 2039–2048.
7. Lee R., Zhou M. Extending PostgreSQL to Support Distributed/Heterogeneous Query Processing. Proceedings of the 12th International Conference on Database Systems for Advanced Applications, April 9-12, 2007, Bangkok, Thailand. – Proceedings. Lecture Notes in Computer Science, Springer, 2007. Vol. 4443. P. 1086–1097.
8. MySQL Cluster Information.
URL: http://www.mysql.com/products/cluster/resources.html (accessed: 10.03.2013).
9. Oracle Store. URL: http://shop.oracle.com (accessed: 10.03.2013).
10. Paes M., Lima A.A.B., Valduriez P., Mattoso M. High-Performance Query Processing of a Real-World OLAP Database with ParGRES. High Performance Computing for Computational Science – VECPAR 2008: 8th International Conference, June 24–27, 2008, Toulouse, France. – Revised Selected Papers. Springer, 2008. P. 188–200.
11. Paulson L.D. Open Source Databases Move into the Marketplace. Computer, – 2004. – Vol. 37, No. 7. – P. 13–15.
12. Pavlo A., Rasin A., Madden S., Stonebraker M., DeWitt D., Paulson E., Shrinivas L., Abadi D.J. Comparison of Approaches to Large Scale Data Analysis. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, June 29 – July 2, 2009, Providence, Rhode Island, USA. ACM, 2009. P. 165–178.
13. Press Release EMC2. URL: http://www.emc.com/about/news/press/2012/20121211-01.htm (accessed: 10.03.2013).
14. Ronstrom M., Oreland J. Recovery Principles in MySQL Cluster 5.1. Proceedings of the 31st International Conference on Very Large Data Bases, August 30 – September 2, 2005, Trondheim, Norway. ACM, 2005. P. 1108–1115.
15. Teradata Purpose-Built Platform Pricing.
URL: http://www.teradata.com/t/WorkArea/DownloadAsset.aspx?id=4682 (accessed: 10.03.2013).
16. Top500 List. URL: http://www.top500.org (accessed: 10.03.2013).
17. VoltDB Documentation. URL: http://voltdb.com/community/documentation (accessed: 10.03.2013).
18. Kuznetcov S.D. MapReduce: vnutri, snarugi ili sboku ot parallelnih SUBD? [MapReduce: inside, outside or beside parallel DMBS]. Trudi Instituta sistemnogo programmirovania RAN [Proceedings of the Institute of System Programming]. 2010. Vol. 19. P. 35–70.
19. Lepikhov A.V., Sokolinsky L.B. Query Processing in a DBMS for Cluster Systems // Programming and Computer Software. – 2010. – Vol. 30, No. 4. – P. 205–215.
20. Pan C.S., Zymbler M.L. Razrabotka parallelnoj SUBD na osnove posledovatelnoj SUBD PostgreSQL s otkrytym ishodnym kodom [Development of a Parallel Database Management System on the Basis of Open-Source PostgreSQL DBMS]. Vestnik Yuzho-Uralskogo gosudarstvennogo universiteta. Seriya "Matematicheskoe modelirovanie i programmirovanie" [Bulletin of South Ural State University. Series: Mathematical Modeling, Programming & Computer Software]. 2012. No. 18(277). Vol. 12. P. 112–120.

Поступила в редакцию 14 июня 2013 г.