Год эпохи перемен в технологии баз данных

Кузнецов С.Д.
kuzloc@ispras.ru

Не дай Вам Бог жить в эпоху перемен
Древняя китайская мудрость

Аннотация. В 2007 г. Майкл Стоунбрейкер и его коллеги опубликовали несколько статей, в которых убедительно утверждали о необходимости радикального пересмотра архитектур СУБД. Эти утверждения подтверждались экспериментами с несколькими новыми архитектурами, предназначенными для поддержки приложений потоковых данных, анализа данных, оперативной обработки транзакций и т.д. В 2008 г. эти тенденции были подтверждены в Клермонтском отчете о направлениях исследований баз данных. Наиболее интересные публикации 2009 г., краткому обсуждению которых посвящена статья, свидетельствуют о том, что идея пересмотра архитектур СУБД всерьез воспринята серьезными исследователями и разработчиками средств управления данными.

1. Введение

Года три назад в мире управления данными возникло течение, авторитетные представители которого утверждали о необходимости радикальных перемен; о том, что «безразмерные» архитектуры универсальных систем управления данными, возраст которых измеряется десятками лет, не в состоянии удовлетворять потребности многих новых приложений; что наступает новая эпоха архитектурно упрощенных специализированных систем управления данными, предельно эффективно и экономично поддерживающих приложения соответствующих достаточно узких предметных областей. Больше всех было слышно Майкла Стоунбрейкера (Michael Stonebraker), его соратников и учеников, ряд экспериментальных и коммерческих разработок которых подтверждал эти утверждения. Однако тогда доводы Стоунбрейкера и Ко о наличии революционной ситуации в области управления данными и потребности в коренном отказе от традиционных архитектур СУБД вызывали сомнения как у независимых экспертов, так и (безусловно!) у специалистов компаний, производящих универсальные СУБД.

Одним из основных событий 2008 г. явилась очередная встреча ведущих специалистов в области управления данных (из академических и коммерческих кругов), результатом которой явился Клермонтский отчет об исследованиях в области баз данных [18]. В подобных отчетах анализируется текущее положение дел и предлагается программа исследований и разработок на ближайшие годы. В Клермонтском отчете (хотя и гораздо более умеренных выражениях, чем в статьях Стоунбрейкера и его сподвижников) также отмечалась потребность в пересмотре архитектур систем управления данными, а также, в частности, указывалось на потребность исследований архитектур СУБД, предназначенных для поддержки «облачных» приложений.

В 2009 г. появилось несколько публикаций, свидетельствующих о ряде успешных разработок систем управления данными, которые основываются на новых архитектурах. Я переводил и комментировал эти публикации. Однако в целом они демонстрируют некоторую общую тенденцию, которую, по моему мнению, стоит проанализировать специальным образом. Подобной попытке анализа и посвящена данная статья.

В разд. 2 будет кратко рассмотрена недавняя предыстория вопроса: основные идеи статей Стоунбрейкера и Ко, соответствующие исследования и разработки, а также некоторые (особенно важные, по моему мнению) положения Клермонтского отчета. Следующие разделы основаны на отдельных публикациях разных авторов 2009 года. В разд. 3 обсуждаются эксперименты по сравнению эффективности технологий MapReduce и массивно-параллельных систем баз данных. В разд. 4 рассматривается новая архитектура СУБД, предназначенная для поддержки Web-приложений в облачной инфраструктуре. Разд. 5 посвящен обсуждению проблем аналитической обработке больших объемов данных. В разд. 6 описываются основные идеи перспективной системы аналитических баз данных, в которой сочетаются возможности создания аналитических приложений на основе SQL и MapReduce. В разд 7 кратко характеризуется новый проект, направленный на создание свободно доступной системы баз научных данных. Наконец, в заключительном восьмом разделе подводятся итоги анализа и приводятся заключительные замечания.

2. Недавняя предыстория

Здесь действительно приходится говорить о совсем короткой предыстории, которая началась для меня только в конце 2006-го-начале 2007 гг.

2.1. Предсказание конца эпохи

В 2006-2008 гг. разными авторами, в число которых почти неизменно входил Майкл Стоунбрейкер, было опубликовано несколько статей, в которых с разной степенью убедительности утверждалось о необходимости радикальных перемен в области управления данными. Все эти статьи переводились на русский язык и комментировались автором данной статьи [1-4].

Комментариям и замечаниям по поводу этих статей была посвящена моя небольшая заметка [5]. Цитируя (с небольшими изменениями) эту заметку, можно отметить, что основные положения указанных статей состоят в следующем:

- Архитектура современных SQL-ориентированных СУБД появилась более 30 лет тому назад, когда рынок систем управления данными был единым, не фрагментированным на специализированные секторы. В это время СУБД вынужденно делались «безразмерными», пригодными для использования в любой области приложений баз данных. Эта «безразмерность» присутствует сегодня в продуктах основных поставщиков. Плюсами основных SQL-ориентированных СУБД является надежность и общая высокая производительность. Минусы – сложность, объемность и высокие накладные расходы, свойственные универсальности.
- За прошедшие 30 с лишним лет рынок систем управления данными сильно фрагментировался. Стали известными большие секторы рынка, для которых очень существенна высокая производительность приложений, которая не достигается или достигается с недопустимо большими затратами при использовании «безразмерных» СУБД. Экономически целесообразной стала разработка специализированных систем, которые ориентируются на эффективную поддержку заранее известных сценариев использования.
- За эти же тридцать лет в области управления данными была выполнена громадная исследовательская работа, результаты которой можно успешно применять для разработки специализированных систем.
- В связи с быстро меняющимися требованиями рынка успешными могут быть только такие новые продукты, которые можно вывести на рынок достаточно быстро – через год или два после начала разработки. Это еще один довод в пользу специализированных систем, нацеленных на решение узкого класса задач. В таких системах требуются более простые языковые средства, упрощается оптимизация запросов и другие аспекты, являющиеся традиционным камнем преткновения «безразмерных» систем.

В последние десять лет Стоунбрейкер последовательно воплощает в жизнь эти идеи. Пробным шаром явилась разработка специализированных средств управления потоковыми данными. На основе исследований и разработок, выполненных в ряде университетов США, была создана компания и промышленная система StreamBase [6], которая была хорошо принята финансовыми компаниями с Уолл-Стрит. На этом этапе Стоунбрейкер практически не конкурировал с «безразмерными» СУБД, для которых рыночный сектор потоковых данных, по-видимому, был слишком узок (кстати, в настоящее время ситуация, похоже, изменилась: в том числе, можно отметить появление компонента Streaminsight [7] в составе ожидаемого релиза Microsoft SQL Server V2).

Следующая попытка Стоунбрейкера состояла в создании нового SQL-ориентированного средства поддержки хранилищ данных с хранением данных по столбцам. И в этом случае созданная компания и промышленная система Vertica [8] основывается на предыдущих университетских исследованиях и разработках, которые, в свою очередь, опираются на многолетние работы других исследователей. Понятно, что в этом случае Стоунбрейкер уже начинает потенциально конкурировать как с «безразмерными» СУБД, так и с СУБД, изначально ориентированными на поддержку хранилищ данных. И в одной из статей приводятся результаты тестовых испытаний, показывающие, что в некоторых сценариях использование приложения, основанного на использовании Vertica, демонстрирует производительность, на два порядка более высокую, чем при использовании «безразмерной» коммерческой СУБД.

Наконец, теперь Стоунбрейкер полностью выходит на тропу войны с «безразмерными» СУБД, покушаясь на их основной, традиционный сектор рынка – OLTP. В исключительно интересном, пока еще университетском проекте H-Store [9] получены результаты испытаний разрабатываемой системы на эталонном тестовом наборе TPC-C, демонстрирующие превосходство над «безразмерной» коммерческой СУБД почти на два порядка.

Кроме того, в одной из статей приводится краткая характеристика и показатели производительности экспериментальной системы ASAP, ориентированной на поддержку математических баз данных. Результаты тоже впечатляют, хотя опубликованных данных относительно общей организации и интерфейсов системы явно недостаточно, чтобы можно было хорошо понять принципы ее организации. Впрочем, это явно будет исправлено при разработке системы SciDB [10] (см. разд. 7), которая основывается на ASAP.

В 2007 г. у меня вызывали определенные сомнения как доводы Майкла Стоунбрейкера и его коллег о целесообразности и даже необходимости полномасштабного отказа от традиционных архитектур СУБД, так и некоторые технические аспекты конкретных систем и их характеристики, описываемые в упомянутых выше статьях. В частности, мне казалось, что к настоящему времени накоплено так много технологических возможностей, как аппаратных, так и программных, что большинство пользователей и разработчиков предпочтет потратить больше средств и/или пожертвовать некоторой долей производительности, чем перейти к использованию радикально других программных средств. Удачный опыт внедрения принципиально новых средств управления потоковыми данными не опровергает эти соображения. В этом случае речь идет об области приложений, которую принципиально не удовлетворяло существовавшее положение дел. Отчасти я был согласен и с патриархальными взглядами Марка Ривкина, высказанными им в статье [11].

2.2. Клермонтский отчет

Раз в несколько лет ведущие представители исследовательского и производственного сообщества баз данных проводят встречи, которые обычно длятся два дня. На этих встречах обсуждается и оценивается состояние дел в области баз данных и формулируются темы исследований, которые будут наиболее актуальны в ближайшие годы. По результатам встреч принято подготавливать и публиковать отчет. Такие отчеты пользуются высоким авторитетом в сообществе баз данных и оказывают серьезное влияние на развитие исследований и разработок. Все опубликованные отчеты доступны в Internet, для многих имеются переводы или пересказы на русском языке [12-17].

В мае 2008 г. состоялась новая встреча. Она проходила в отеле Claremont Resort в г. Беркли, штат Калифорния, и поэтому отчет получил название «Клермонтского». Джо Хеллерстейн (Joseph M. Hellerstein) подготовил и поддерживает специальный сайт (http://db.cs.berkeley.edu/claremont/), на котором и был опубликован первый вариант отчета [18]. Имеется его полный перевод на русский язык.

Я не буду здесь подробно пересказывать отчет, а лишь отмечу те его места, которые имеют отношение к дальнейшему материалу этой статьи.

2.2.1 Проблемы архитектур серверов баз данных

В качестве одной из наиболее актуальных тем исследований отмечается *пересмотр архитектуры серверов баз данных*. В отчете утверждается (здесь и ниже текст цитируется не вполне точно), что

«…у современных развитых коммерческих систем реляционных баз данных имеются хорошо известные ограничения. Обеспечивая широкий набор возможностей, они показывают пиковую производительность только для очень ограниченного набора режимов. В последнее десятилетие появилось много популярных задач, связанных с обработкой больших объемов данных, для которых реляционные СУБД обеспечивают плохое соотношение «цена/производительность», и при решении которых от использования РСУБД пришлось отказаться.»

Как видно, эти утверждения напрямую перекликаются с доводами Майкла Стоунбрейкера и его сторонников, рассмотренными в предыдущем подразделе.

С другой стороны, в отличие от общего «революционного» тона статей Стоунбрейкера и Ко, в отчете говорится, что

« имеются два разных направления: расширение диапазона применимости универсальных систем баз данных и радикальное повышение производительности путем разработки специализированных систем баз данных для конкретных прикладных областей. У обоих направлений имеются свои достоинства, и очевидная общность конечных целей подсказывает, что

работы в этих направлениях можно выполнять с взаимной пользой: специализированные методы можно повторно использовать в более универсальных системах, а использование архитектурных компонентов универсальных систем может позволить быстрее создавать прототипы новых специализированных систем.»

На самом деле, так и происходит. Наиболее развитые традиционные СУБД непрерывно развиваются за счет внедрения технологий, разработанных для специализированных (часто экспериментальных) систем (заметим, что это отнюдь не снижает их уровень сложности), а при разработке специализированных систем с новой архитектурой грамотные разработчики стремятся к максимальному использованию проверенных методов.

К числу наиболее важных исследовательских тем в этой области относятся:

1. разработка систем для кластеров многоядерных процессоров, в которых имеется ограниченный и неоднородный доступ к памяти вне кристалла;

2. использование удаленной основной и флэш-памяти в качестве среды персистентного хранения данных в дополнение к памяти на магнитных дисках;

3. разработка унифицированного подхода к постоянно выполняемой адаптации и самонастройке оптимизации запросов и физических структур хранения данных;

4. сжатие и шифрование данных на уровне хранения, интегрированное со структурой хранения и оптимизацией запросов;

5. разработка систем, опирающихся на не реляционные модели данных, вместо того, чтобы «впихивать» эти данные в таблицы;

6. нахождение компромиссов между согласованностью и доступностью для достижения лучшей производительности и масштабности уровня тысяч машин;

7. разработка СУБД, учитывающих потребление энергии, которые ограничивают энергопотребление без ущерба для масштабируемости.

Для целей обсуждения в этой статье основной интерес в этом списке представляет п. 6. Из остальных наиболее актуальным мне кажется п.2. В этой статье он подробно не обсуждается, но совершенно очевидно, что бурное развитие технологии флэш-памяти и «твердотельных дисков», доступность таких носителей с емкостью в сотни гигабайт создает новые перспективы для организации систем управления данными.

В связи с этим на меня произвела большое впечатление статья Гоца Грейфа «Правило пяти минут двадцать лет спустя, и как флэш-память изменяет правила» [19]. В этой статье автор, в частности, задает вопрос:

«Смогут ли традиционные системы, в которых вместо традиционных дисков применяется флэш-память, конкурировать со специализированными системами баз данных в основной памяти по производительности, общей стоимости владения, стоимости разработки и сопровождения, времени выхода на рынок и выпуска очередных релизов и т.д.?»

Другими словами, имеются основания предполагать, что применение флэш-памяти может существенно изменить как архитектуру, так и номенклатуру систем управления данными.

2.2.2 Декларативное программирование

Следующей важной темой исследований, выделяемой в Клермонтском отчете и имеющей отношение к данной статье, является *декларативное программирование для новых платформ*. В отчете говорится, что

«…актуальность этой проблемы возрастает буквально экспоненциально, поскольку программисты нацеливаются на все более сложные среды, включая многоядерные микропроцессоры, распределенные службы и платформы «облачных вычислений» (cloud computing). Неквалифицированным программистам требуется возможность легко писать надежные программы, масштабируемые при возрастании числа процессоров как в слабосвязанных, так и в сильносвязанных архитектурах».

Авторы отчета полагают, что подходы, облегчающие и упрощающие программирование приложений, которые ориентированы на обработку данных, окажут серьезное влияние на программирование в целом.

В качестве первого примера подобного подхода рассматривается MapReduce. Отмечается, что:

«…в подходе Map-Reduce привлекает его простота; он основывается на языковых методах и методах распараллеливания по данным, известных в течение десятилетий. Для исследователей баз данных значимость подхода Map-Reduce состоит в том, что он демонстрирует преимущества программирования с распараллеливанием по данным новым классам разработчиков».

Со своей стороны замечу, что этот подход действительно чрезвычайно популярен, особенно среди молодежи. С другой стороны, как показывают исследования, обсуждаемые в разд. 3, очень важно не переоценивать этот подход и относиться к нему, как к первому шагу на пути упрощения параллельной обработки данных. Интересно также отметить лояльное отношение в MapReduce среди некоторых разработчиков новых систем баз данных. В связи с этим см. разд. 6.

Вторым примером является появление новых декларативных языков, основанных на Datalog. В ряде сценариев использование подобного декларативного языка позволяет на порядки величин сократить объем кода, одновременно обеспечивая его распределенное или параллельное исполнение. Как отмечается в отчете, группы, выполняющие соответствующие исследования, очень слабо скоординированы, и возрождение декларативных языков в новых контекстах пока происходит практически стихийным образом. Тем не менее, сам факт реинкарнации Datalog кажется мне крайне интересным и потенциально многообещающим.

Наконец, третий пример происходит из области программирования корпоративных приложений. Сближению сред программирования и управления данными, устранению проблемы потери соответствия (impedance mismatch) способствует возникновение новых языковых расширений, таких как LINQ [20] и Ruby on Rails [21]. Однако в отчете отмечается, что:

«… в этих пакетах пока серьезно не решается проблема программирования с использованием нескольких машин. Для корпоративных приложений ключевым решением распределенной разработки является разделение логики приложения и данных между несколькими «звеньями» («tier»): Web-клиентами, Web-серверами, серверами приложений и серверной СУБД. Особо ценной здесь является независимость данных, позволяющая создавать программы без предварительного принятия долговременных решений о физическом размещении программ и данных в разных звеньях».

Как полагают авторы отчета, одним из существующих языков, который мог бы способствовать подобному декларативному программированию, является XQuery, поскольку формат XML часто используется на разных уровнях протоколов. Практическим подтверждением справедливости этого предположения является работа, рассматриваемая в разд. 4.

2.2.3 Управление данными в «облачной» инфраструктуре

Как отмечается в отчете, облачные сервисы могут повысить эффективность поставщиков приложений за счет сокращения объема требуемых начальных вложений и сокращения стоимости владения. Доступен ряд разнообразных облачных сервисов, включая прикладные сервисы (salesforce.com [22]), сервисы хранения (Amazon S3 [23]), вычислительные сервисы (Google App Engine [24], Amazon EC2 [25]) и сервисы данных (Amazon SimpleDB [26], Microsoft SQL Server Data Services [27], Google’s Datastore [28]).

Общей проблемой поставщиков облачных сервисов является потребность в компромиссе между функциональными возможностями и эксплуатационными расходами. Интерфейсы существующих облачных служб намного более ограничены, чем в традиционных системах баз данных. В них поддерживаются минималистские языки запросов и ограниченные гарантии согласованности данных. Это затрудняет программирование приложений, но позволяет создавать более предсказуемые службы, в которых обеспечиваются

соглашения об уровне обслуживания, трудно достижимые для полнофункциональных служб данных на основе языка SQL.

Следующая проблема – достижения высокого уровня управляемости. Решение этой проблемы для систем в облачной среде осложняется ограниченным человеческим вмешательством, сильно изменчивыми рабочими нагрузками и разнообразием совместно используемых инфраструктур. При решении проблемы требуется учитывать наличие различных подходов к виртуализации ресурсов – от виртуализации аппаратных средств до использования в службе данных одной СУБД для управления несколькими базами данных с разными схемами. Для достижения управляемости требуется развивать и совершенствовать методы самоуправления системами.

Серьезную проблему представляет масштабируемость будущих облачных систем управления данными. Технология, применяемая при разработке современных SQL-ориентированных СУБД, просто не позволяет масштабировать их до 1000 узлов. По всей видимости, в будущих системах нужно ограничивать требования к их транзакционности или учитывать семантику данных. Дополнительные исследования нужны для понимания реальной масштабируемости облачной инфраструктуры. В связи с этим см. разд 3 и 4 настоящей статьи.

Наконец, при совместном использовании физических ресурсов в облачной инфраструктуре требуется обеспечение безопасности и конфиденциальности данных, которые не могут гарантироваться за счет наличия физического разграничения машин или сетей.

Рассмотрим теперь, как утверждения, предположения и прогнозы, рассмотренные в этом разделе, проявляются в работах, опубликованных в 2009-м году.

3. MapReduce и параллельные системы баз данных

Молодежи свойственно увлекаться новыми идеями. Идея MapReduce [29], выдвинутая и реализованная сначала Google, а потом и сообществом open source в проекте Hadoop [30] почти мгновенно овладела молодыми массами. Причем даже теми представителями компьютерной молодежи, которые получили хорошее образование и последующий практический опыт в области систем управления базами данных.

Мне неоднократно приходилось слышать от молодых коллег, что они считают достоинствами MapReduce отсутствие схемы данных (в том числе, и отсутствие поддержки типов данных) и даже потребность в явном программировании конструкций, которые испокон веков поддерживались в СУБД на уровне высокоуровневых языковых конструкций языка SQL. Дополнительным стимулом к применению MapReduce была привязка этой технологии к «облачным» вычислениям, возможность практически бесплатно арендовать виртуальный кластер с большим числом узлов и развернуть на нем

свою MapReduce программу, почти автоматически достигнув громадной производительности своего приложения.

До поры до времени представители старшего и среднего поколений сообщества баз данных ограничивались ворчанием в адрес MapReduce, что, в свою очередь, еще больше привлекало молодежь к использованию соответствующих средств. Действительно, раз «старики» ворчат, значит, они просто не понимают, что средства управления данными их поколений просто устарели и нужно переходить к использованию новых, прогрессивных технологий. Больше других ворчали (и получали достойную отповедь от молодежи) Майкл Стоунбрейкер и Дэвид Девитт (David J. DeWitt), что, в частности, проявилось в написании и публикации ими в январе 2007 г. в коллективном блоке Database Column [31] заметок [32-33].

И вот, в 2008 г. ворчание стариков выразилось в инициировании ими чрезвычайно интересного проекта по практическому сравнению технологии MapReduce с технологиями параллельных СУБД категории sharing nothing. Результатам этого проекта посвящена статья [34].

3.1. Эксперименты и их результаты

Эксперименты, описываемые в этой статье, основывались на выполнении нескольких несложных аналитических задач на кластере из ста узлов с использованием двух параллельных СУБД категории sharing-nothing (применялись СУБД Vertica и некоторая СУБД с традиционным построчным хранением данных, именуемая в статье СУБД-Х) и реализации MapReduce от Hadoop. Очень интересно, как авторы обосновывают свой выбор для экспериментов относительно небольшого кластера, тогда как сторонники MapReduce обычно говорят об абсолютных преимуществах этой технологии при использовании кластеров с тысячами узлов.

Как утверждается в статье, такая масштабная аппаратура не требуется современным высокоэффективным параллельным СУБД даже при поддержке предельных в настоящее время по объему петабайтных баз данных. Например, в конфигурации СУБД Teradata, применяемой в eBay, 72 узлов (в каждом узле два четырехъядерных процессора, 32 гигабайта основной памяти и 104 300-гигабайтных диска) оказывается достаточно для управления реляционными данными объемом около 2,4 петабайт. Так что с самого начала статьи, по сути дела, утверждается, что сверхвысокая масштабируемость параллельных СУБД в действительности не требуется (заметьте, как это перекликается с тезисом Клермонтского отчета о невозможности должного масштабирования современных СУБД в облачной инфраструктуре).

При том, что авторы статьи уделили тщательное внимание программированию тестовых аналитических задач для среды MapReduce, на кластере из 100 узлов СУБД-Х показала среднюю производительность, большую соответствующего варианта для MapReduce в 3,2 раза, а СУБД Vertica оказалась в 2,3 раза быстрее СУБД-Х. Авторы предполагают, те же относительные соотношения

сохранились бы и на кластере с 1000 узлами, хотя такое оборудование просто не требуется для поддержки параллельными системами баз данных актуальных сегодня баз данных петабайтного масштаба.

3.2. В чем причины?

Преимущество в производительности, демонстрируемое параллельными СУБД, оказывается возможным за счет использования технологий, совершенствуемых в течение десятилетий:

- индексация на основе B-деревьев для оптимизации выборки данных;
- новые механизмы хранения данных (в частности, поколоночное хранение);
- эффективные методы сжатия данных, позволяющие выполнять операции прямо над сжатыми данными;
- параллельные алгоритмы выполнения операций над реляционными данными.

С другой стороны, при использовании MapReduce потребовалось существенно меньшие времени на загрузку данных: иногда это происходило в три раза быстрее, чем в случае Vertica, и 20 раз быстрее, чем для СУБД-X (в основном, из-за отсутствия схемы данных). Существенно проще происходит установка и конфигурирование MapReduce, чем аналогичные действия для параллельных СУБД. Авторы статьи приходят к выводу, что это является одним и основных факторов, привлекающих к MapReduce разработчиков приложений. Другими факторами является простота использования (SQL часто оказывается слишком сложным для начинающих программистов аналитических приложений; с другой стороны примитивность MapReduce приводит к росту трудозатрат на разработку приложений) и большая устойчивость к сбоям отдельных узлов кластера.

Принципиальным отличием MapReduce от систем баз данных является отсутствие схемы. В результате возникает потребность разбора записей на стадии выполнения приложения, уменьшается польза от сжатия данных. Все это является одной из причин снижения производительности. Кроме того, при отсутствии схемы невозможно обеспечить унифицированное хранение информации, требуемой для оптимизации декларативных запросов: об имеющихся индексах, о применяемых способах разделения таблиц, о гистограммах, описывающих распределения данных и т.д.

Общим выводом статьи является потребность в совершенствовании обеих технологий. MapReduce нуждается в более выразительных средствах программирования, чтобы можно было снизить упомянутые выше трудозатраты на разработку. В параллельных СУБД требуются более совершенные методы распараллеливания определяемых пользователями функций. Первыми признаками грядущей интеграции SQL и MapReduce являются системы Greenplum [35] (см. разд. 6) и *n*Cluster компании Aster Data [36]. Хочу также заметить, что в августе 2009 г. поддержка MapReduce

появилась и в компании Vertica [37], являющейся, как уже говорилось, детищем ранее неутомимого борца с этим подходом Майкла Стоунбрейкера.

4. Cloud Computing и новые критерии оптимальности архитектуры СУБД

Хотя «облачные» вычисления (cloud computing) постепенно входят в практику все большего числа разработчиков приложений, остается открытым вопрос, *какими должны быть средства управления данными «в облаках»*? Некоторые люди полагаются на использование распределенных файловых систем и подхода map/reduce, другие говорят о целесообразности использования массивно-параллельных систем баз данных (см. разд. 2), третьи пытаются приспособить к облачной инфраструктуре традиционные SQL-ориентированные СУБД (например, SQL Azur [38] от Microsoft; кстати, после появления в Microsoft проекта массивно-параллельной СУБД Madison (в настоящее время соответствующий продукт называется SQL Server 2008 R2 Parallel Data Warehouse [39]), возможно, компания присоединится к лагерю параллельных облачных СУБД).

Свой подход к «облачным» СУБД предлагают Даниела Флореску (Daniela Florescu) и Дональд Коссман (Donald Kossmann) в статье [40]. В качестве краткой справки заметим, что эти авторы наиболее известны своим активным участием в разработке стандарта языка XQuery и другими работами, связанными с языком XML.

Статья, по сути, состоит из двух основных частей. В первой части вводятся и обосновываются новые критерии оптимальности СУБД, соответствующие современной специфике (наличию облачной инфрастуктуры и крупных центров данных с дешевыми аппаратными средствами). Во второй части предлагается и объясняется архитектура СУБД, соответствующая этим критериям.

4.1. Новые критерии оптимизации

Традиционные критерии оптимизации, по мнению авторов, выглядят следующим образом:

> При заданном наборе аппаратных ресурсов и гарантировании полной согласованности данных (т.е. при поддержке ACID-транзакций) требуется минимизировать время ответа за запросы и максимизировать пропускную способность системы по отношению к поступающим запросам.

Они считают, что в новых условиях они должны быть такими:

> При заданных требованиях к производительности приложений (пиковая пропускная способность, предельно допустимое время ответа) требуется минимизировать требуемые аппаратные ресурсы и максимизировать согласованность данных.

Основным показателем, нуждающемся в оптимизации, является оценка денежных *затрат*. Наиболее важный вопрос состоит в том, сколько требуется машин, чтобы удовлетворить требования к производительности на заданной рабочей нагрузке. Аппаратные ресурсы – это теперь не одноразовая инвестиция; это отдельная существенная статья ежемесячных расходов IT.

Для большинства приложений *производительность* является заданным ограничением, а не целью оптимизации. Требуется, чтобы поддерживалась заранее заданная пиковая рабочая нагрузка. Имеется возможность поддержки, по существу, любой рабочей нагрузки, вопрос только в том, сколько это будет стоить, и чем дешевле, тем лучше.

Насущной потребностью любой современной информационной системы является неограниченная *масштабируемость*. Это требование удовлетворяется, если расходы на IT растут линейно по мере роста бизнеса, и этот рост ничем не ограничивается. У традиционных СУБД функция стоимости является ступенчатой, а масштабируемость – ограничена.

Для многих организаций *предсказуемость* приложений настолько же обязательна, что и масштабируемость. Имеется в виду предсказуемость и производительности, и стоимости. Оптимизация 99% операций более важна, чем оптимизация в среднем. Прилагаются огромные усилия для сокращения расходов на администрирование и достижения более устойчивой производительности СУБД, но радикальных успехов достичь до сих пор не удалось.

Транзакции в стиле ACID, обеспечивающие полную *согласованность* данных, и сервис-ориентированная архитектура, помогающая структурировать и развивать приложения, плохо совместимы. Более того, большинству приложений просто не требуется полная согласованность. Более приоритетным является требование стопроцентной доступности данных. В современных приложениях согласованность данных является целью оптимизации, а не ограничением, как в традиционных системах баз данных.

Гибкость – это возможность настройки программной системы к индивидуальным требованиям пользователя. По мере усложнения приложений гибкость как цель разработки становится все более важной. В традиционных системах OLTP это требование отсутствовало, так что гибкость не являлась целью оптимизации в традиционных системах управления данными. Понятно, что гораздо проще обеспечить должные производительность и масштабируемость, если система не обладает гибкостью.

4.2. Новая архитектура СУБД

В традиционной архитектуре СУБД (рис. 1) все запросы инициируются пользователями на уровне представления, например, с использованием Web-браузера. Логика приложения программируется на среднем уровне; на среднем же уровне поддерживаются такие функциональные средства, как

Web-сервер. Все управление данными производится на низшем уровне с использованием СУБД.

Рис. 1. Традиционная архитектура СУБД

Эта архитектура практически идеально отвечает критериям оптимальности традиционных СУБД. Согласованность данных поддерживается СУБД на нижнем уровне. Основная цель оптимизации (минимизация времени отклика и максимизация пропускной способности) достигается за счет применения на всех уровнях ряда отработанных методов (индексация, кэширование, разделение и т.д.). Верхние два уровня почти неограниченно масштабируются (масштабируемость нижнего уровня ограничена).

Однако целям новой оптимизации традиционная архитектура не удовлетворяет. Во-первых, она рассчитана на обеспечение ожидаемой пиковой производительности, которая может на несколько порядков превышать реальную среднюю производительность. В результате б*о*льшая часть дорогостоящих аппаратных ресурсов простаивает. То же можно сказать о многих компонентах программного обеспечения СУБД, за которые приходится платить даже в том случае, когда они не требуются приложению. Во-вторых, традиционная архитектура не обеспечивает предсказуемость, поскольку при наличии многопользовательского доступа трудно понять, что происходит на уровне СУБД. В-третьих, как уже отмечалось, на нижнем уровне архитектуры масштабируемость ограничена. Наконец, не поддерживается гибкость, поскольку на всех трех уровнях используются разные модели данных и программирования (XML/HTML и скриптовые языки

на верхнем уровне, объектно-ориентированные подход на среднем уровне и SQL – на нижнем уровне).

Двумя принципами разработки приложений в трехзвенной архитектуре являются *контроль* и *передача запросов*. Первый принцип означает, что весь доступ к данным полностью контролируется СУБД, а второй – что как можно большая часть логики приложений должна выполняться поблизости от данных, т.е. внутри СУБД (на основе хранимых процедур, определяемых пользователями типов данных и функций), По мнению авторов, эти два принципа подрывают масштабируемость, предсказуемость и гибкость, приводят к удорожанию и усложнению СУБД (хотя вполне согласуются с целями традиционной архитектуры).

Предлагаемая авторами архитектура показана на рис. 2. Здесь на нижнем уровне поддерживается только крупномасштабное распределенное хранение данных на основе, например, облачной службы Amazon S3. Согласованность данных обеспечивается на среднем уровне, причем эта согласованность не является строгой. Она поддерживается за счет применения всеми серверами приложений общих соглашений при чтении и записи данных через службу нижнего уровня.

Рис. 2. Новая архитектура

На всех уровнях можно использовать дешевую аппаратуру и обеспечивать неограниченную масштабируемость. На каждом уровне допускается выход из строя любого узла: на нижнем уровне это возможно из-за репликации и слабой согласованности данных; на верхних – за счет работы узлов без сохранения состояния.

На основе этой новой архитектуры в компании 28msec [41] реализуется система Sausalito. В качестве службы хранения данных используется Amazon S3, на среднем уровне применяются сервисы прикладного уровня Amazon EC2. Вся традиционная функциональность управления базами данных реализуется на этом же уровне на основе языка XQuery с расширениями, обеспечивающими возможность модификации данных (понятно, что в данном случае речь идет об XML-данных). Тот же язык используется для разработки Web-приложений, на поддержку которых и ориентирована система Sausalito.

4.3. Все ли так хорошо с новой архитектурой?

На мой взгляд, статья написана очень последовательно и логично. Правда, несколько смущает сходство предлагаемой архитектуры приложений баз данных с архитектурами файл-серверных СУБД. Фактически, в подходе Флореску и Коссмана Amazon S3 выполняет роль файл-сервера, а вынесение службы запросов и других функций СУБД на уровень приложения напоминает организацию систем, существовавших до появления клиент-серверных архитектур, например, Informix SE.

И, конечно, что там не говори насчет отказа от принципа «передачи запросов», некоторые сомнения вызывает передача по Internet от узлов Amazon S3 в узлы серверов приложений, как минимум, XML-документов целиком (а может быть и коллекций XML-документов). Непонятно, как при этом удается гарантировать, что время ответа на запрос не превышает заданные ограничения (если, конечно, не считать, что пользователи могут спокойно подождать и несколько минут).

Кроме того, я не уверен, что разработчики приложений придут в полный восторг от необходимости использования XQuery не только для запросов XML-данных, но и для написания логики приложений. Для системы это, конечно, очень удобно, а на месте разработчиков я бы, пожалуй, предпочел использовать для программирования что-нибудь более привычное.

Хотя, конечно, здесь нужно учитывать, что Флореску и Коссман входят в число родоначальников XQuery, и с их точки зрения, по-видимому, программирование приложений на XQuery является вполне естественным и удобным. Кстати, по этому поводу полезно почитать статью [42].

А может быть, я напрасно сомневаюсь, этот возврат к архитектурному прошлому закономерен, очень большие XML-документы в Web-приложениях не встречаются, а разработчики Web-приложений с энтузиазмом перейдут от программирования на PHP и Python к программированию на XQuery. Авторы достаточно убедительно демонстрируют, что их «новая» архитектура (а любое

«новое», как известно, – это хорошо забытое «старое») обеспечивает неограниченное масштабирование приложений, гибкость системы и предсказуемость ее поведения. В конце концов, в старые и добрые времена файл-серверных СУБД не было облачных вычислений, не было Web-приложений и требования к системам баз данных были другими.

5. Как справиться с большими данными?

В области управления данными ежегодно публикуется множество статей – в сборниках трудов многочисленных конференций, в специализированных журналах и изданиях универсальной софтверной тематики. Однако подавляющее большинство этих статей касается чрезвычайно узкой тематики, понятной только специалистам, профессионально занимающимся аналогичными вопросами. Лишь немногие люди решаются публично выразить свою более общую точку зрения, затрагивающую проблемы области в целом. И круг таких людей очень ограничен: Майкл Стоубрейкер, раньше – Джим Грей (Jim Gray), который, к всеобщему несчастью, пропал в океане зимой 2007 г. [43], Кристофер Дейт (Chris Date), может быть, еще несколько человек. Новые люди в этом «элитном» сообществе появляются крайне редко.

Поэтому меня сразу заинтересовала статья Адама Якобса «Патологии больших данных» [44]. Привлекли амбициозное название статьи, а также неизвестность и явное нахальство автора, решившегося высказаться на такую тему. Как удалось выяснить, первой специализацией автора статьи была лингвистика, а степень PhD он получил в области вычислительной нейропсихологии. Ему приходилось заниматься аналитическими исследованиями больших объемов данных, и с начала 2000-х он работает в компании 1010data Inc. [47], где руководит разработкой аналитической СУБД Tenbase [46].

Статья Якобса настолько меня затронула, что я написал по ее поводу отдельную заметку «О точности диагностики патологий» [45], в которой серьезно (и, полагаю, заслуженно) раскритиковал автора. Не буду здесь пересказывать эту заметку, а остановлюсь только на том, что связывает статью Адама Якобса с основной темой моей статьи.

На основе своих рассуждений Якобс приводит следующее «метаопределение» больших данных:

> Большими являются данные, размер которых вынуждает нас выходить за пределы проверенных временем методов, широко распространенных в данное время.

В середине 1980-х гг. приходилось иметь дело с наборами данных, которые были настолько масштабными, что для их обработки требовалось использовать специальные робототехнические решения, автоматизирующими работу с тысячами магнитных лент. В 1990-е гг. для анализа больших данных не хватало возможностей Microsoft Excel и персональных компьютеров, и в

этом случае использовались рабочие станции с ОС Unix и более серьезным программным обеспечением. В настоящее время большими являются такие данные для обработки которых оказываются недостаточными средства традиционных SQL-ориентированных СУБД и настольных статистических пакетов; требуется массивно-параллельное программное обеспечение.

Как видно, по мнению автора, с проблемой больших данных всегда приходится сталкиваться именно при анализе данных; в области оперативной обработки транзакций такой проблемы нет. И решить проблему больших данных в каждый период времени можно только за счет отказа разработчиков программных систем от традиционных, типовых решений, за счет понимания истинной природы имеющихся аппаратных средств и привлечения всего многообразия ранее созданных методов и алгоритмов. Мне кажется, что в этом идеи Якобса полностью созвучны идеям Стоунбрейкера, хотя он вообще на него не ссылается.

Что касается конструктивных предложений, то в статье Якобса я их вижу две:

- несмотря на совершенствование устройств внешней памяти (включая устройства флэш-памяти), их можно эффективно использовать только при последовательном доступе; поэтому следует избегать неоптимальных схем доступа к внешней памяти;
- в распределенных системах баз данных репликация должна служить не только целям повышения уровня доступности и отказоустойчивости системы; для повышения эффективности системы баз данных следует поддерживать реплики данных с разным физическим представлением.

Судя по всему, эти предложения используются при разработке упомянутой выше распределенной СУБД Tenbase [46]. Как я уже отмечал, многие высказывания Якобса заслуживают критики. Еще раз отсылаю читателей к своей заметке [45].

6. Новый взгляд на место аналитиков в системе баз данных

С идеями Стоунбрейкера интересным образом сплетается статья [48]. Главным автором статьи, безусловно, является Джо Хеллерстейн, профессор Калифорнийского университета в Беркли, научный руководитель компании Greenplum [35], в которой работают остальные авторы статьи, соратник Майкла Стоунбрейкера по разработке СУБД Postgres.

В статье утверждается, что традиционный, «ортодоксальный» подход к организации корпоративных хранилищ данных (Enterprise DataWarehouse, EDW), основанный классиками этого направления Эдгаром Коддом (Edgar Codd), Биллом Инманом (Bill Inmon) и Ральфа Кимболла (Ralph Kimball), не соответствует реалиям настоящего времени. В этом традиционном подходе главным является тщательное проектирование и развитие схемы EDW, служащей основой интеграции корпоративных данных. Сервер баз данных,

поддерживающий EDW, является основным вычислительным средством, центральным, масштабируемым механизмом корпоративной аналитики. EDW контролируется специально назначаемыми сотрудниками IT, которые не только сопровождают систему, но и тщательно контролируют доступ к ней.

В настоящее время хранение данных обходится настолько дешево, что за счет собственного бюджета базу данных громадного масштаба может иметь даже небольшое подразделение корпорации. Число внутрикорпоративных источников данных непрерывно возрастает, включая журналы Web-серверов, архивы электронной почты и т.д. Все отчетливее понимается важность аналитики, разной аналитики в разных подразделениях одной и той же компании. Для аналитики нужны как можно более свежие данные, целесообразна поддержка отдельных механизмов сбора и анализа данных для разных подразделений.

6.1. Вся власть – аналитикам!

Исходя из этого изменения ландшафта анализа данных, авторы предлагают подход MAD, являющийся акронимом от *magnetic (магнетичность), agile (гибкость)* и *deep (основательность)*. В совокупности эти три характеристики подхода образуют *MAD skills*, новые, *МОГучие способности* анализа данных.

По мнению авторов, ортодоксальные EDW «отталкивают» новые источники данных: для интеграции в хранилище данных заново возникшего источника данных требуется, вообще говоря, изменить схему EDW, настроить (или даже создать) процедуру ETL (Extract-Transform-Load) и применить эту процедуру. Весь этот процесс может длиться месяцами, а может и вообще не закончиться. В результате аналитики остаются без данных, анализ которых мог бы принести бо́льшую пользу компании. Предлагаемая концепция *магнетичного* хранилища данных означает, что данные должны становиться доступными для анализа сразу после появления нового источника. Аналитики должны сами решать, что для них важнее, полная очищенность и согласованность данных или же быстрота доступа к ним. Диктовать условия доступа к данным должны не администраторы EDW, а пользователи, т.е. аналитики.

Процессы развития хранилища данных, должны быть *быстрыми* и *гибкими.* Должна допускаться возможность быстрого изменения физического и логического содержимого аналитической базы данных. В частности, это означает, что у «могучего» хранилища данных может отсутствовать ортодоксальная жесткая схема (хотя ее могут поддерживать сами аналитики). Может отсутствовать и ортодоксальная процедура ETL: аналитики сами должны решать, какой уровень согласованности данных им требуется.

В современных СУБД поддерживаются только минимальные средства, требуемые для анализа данных (типа CUBE BY). Серьезные статистические пакеты (SAS, Mathlab, R) выполняются на рабочих станциях, что ограничивает объем анализируемых данных и требует их передачи по сети. *Основательность* нового подхода к аналитике означает возможность

разработки самими аналитиками и размещения поблизости от данных, внутри аналитической СУБД статистических пакетов любой сложности. При этом аналитикам не должен навязываться конкретный стиль разработки этих пакетов. Например, им должны быть равно доступны технологии SQL и MapReduce.

6.2. Новые возможности за счет старых средств

В соответствии с подходом MAD в компании Greenplum на основе свободно доступных исходных текстов СУБД Postgres (и PostgreSQL) разработана параллельная аналитическая СУБД, пригодная, в частности, для использования в среде cloud computing. Насколько я понимаю, основные усилия были затрачены на разработку массивно-параллельного варианта системы, поскольку Postgres никогда не была параллельной системой. Естественно, о качестве этой части работы, не испытав ее в реальных производственных условиях, говорить невозможно.

Но зато все остальные расширения (поддержка статистических пакетов вблизи от данных, реализация MapReduce) основываются на использовании стандартных средств Postgres, которая с самого начала разработки в середине 1980-х гг. задумывалась Стоунбрейкером, как расширяемая система. Определяемые пользователями типы данных и функции, возможность определения новых структур хранения и методов доступа, естественно, позволяют сравнительно легко добавлять в систему новые, даже совсем не ожидавшиеся возможности. В связи с этим меня всегда интересовал вопрос, насколько безопасны эти расширения, не могут ли они подорвать общую работоспособность системы? Ответ на этот вопрос в контексте Postgres (и позднее в контексте Informix Universal Server) я получить так и не смог. Не дают его и авторы данной статьи.

В заключение этого раздела отметим сходства и различия подхода Хеллерстейна с подходов Флореску и Коссмана, обсуждавшимся в разд. 4. В обоих случаях критикуется традиционная архитектура, в обоих случаях предлагается некоторая новая архитектура, более пригодная для использования в современных условиях, в частности, на основе инфраструктуры cloud computing. Основным отличием является, на мой взгляд, отношение к принципу «передачи запросов». Флореску и Коссман жертвуют этим принципом ради достижения масштабируемости и предсказуемости (а также в угоду SOA), а Хеллерстейн, по сути, целиком на нем основывается. Конечно, в первом случае речь идет о Web-приложениях обработки данных, а во втором – об аналитике. Но в любом случае идея физического приближения вычислений к данным всегда была притягательной: зачем передавать по сети данные, если можно передавать существенно менее объемные результаты их обработки?

7. Научные базы данных и проект SciDB

У ученых разных специальностей (физиков, химиков, астрономов, социологов и т.д.) исторически существуют сложные взаимоотношения с миром баз данных. Это видно, например, при анализе проектов Российского фонда фундаментальных исследований, связанных с созданием научных информационных систем. Им неудобны главенствующие в мире SQL-ориентированные СУБД, предназначенные, главным образом, для поддержки разных видов бизнеса.

В свое время об этом много думал и писал Джим Грей (см. например, [49]). В его честь по инициативе, прежде всего, Майкла Стоунбрейкера и Дэвида Девитта в начале 2009 г. образован проект SciDB (http://www.scidb.org/). К настоящему времени (начало декабря 2009 г.) по поводу этого проекта опубликованы две статьи: «Requirements for Science Data Bases and SciDB» [50] и «A Demonstration of SciDB: A Science-Oriented DBMS» [51]. На русский язык эти статьи не переводились.

В число основных проектировщиков SciDB, помимо Стоунбрейкера и Девитта, входят, в частности, такие известные в мире баз данных люди, как Сэм Мэдден (Sam Madden), Дэвид Дайер (David Maier), Дженнифер Вайдом (Jennifer Widom) и Стэн Здоник (Stan Zdonik). Разработчиков пока на вид меньше, чем проектировщиков, но среди них российские программисты Павел Велихов и Роман Симаков. Проект выполняется в стиле open source (хотя никаких исходных текстов на сайте проекта пока нет) и рассчитан на два года. Проект поддерживается спонсорами, включая компании Vertica и eBay. Других источников финансирования, похоже, пока нет, хотя в начале проекта говорилось о возможной финансовой поддержке со стороны National Science Foundation.

Как отмечалось в разд. 2, проекту SciDB предшествовал университетский проект ASAP, из которого заимствуются многие идеи. Среди основных характеристик ожидаемой системы на текущий момент можно выделить следующее:

- Используется модель данных, основанная на популярных среди ученых разных специальностей вложенных многомерных массивах.
- Поддерживаются примитивные операции, ориентированные на научные расчеты, такие как смещение координатной сетки.
- Для всех хранимых данных обеспечивается информация об их происхождении, т.е. из какого источника данных они взяты.
- Обеспечивается возможность хранения, выборки и обработки неточных данных.
- Имеется возможность обработки данных без их загрузки в базу данных.

Как видно, проект SciDB вполне соответствует идеям Стоунбрейкера: быстро реализуется специализированная система. Предполагается возможность использования SciDB в облачной инфраструктуре. Насколько этот проект

будет успешным, покажет ближайший год (второй, завершающий год выполнения проекта).

8. Заключение

Как показывает 2009 г., хотим мы этого или не хотим, в мире баз данных происходят серьезные изменения. В той или иной степени все наиболее интересные, на мой взгляд, события 2009-го года связаны с идеями Майкла Стоунбрейкера и Клермонтского отчета, опубликованными в 2007-2008 гг. Специализация систем управления данными, поиск их новых архитектур, более приспособленных к текущим реалиям, становится относительной нормой.

Однако нужно учитывать, что 2009-й год был кризисным. Известно, что кризис – это хорошее время для новых начинаний и плохое время от зрелого бизнеса. Поэтому мы еще не услышали ответ на новые инициативы в области управления данными со стороны компаний, являющихся основными производителями СУБД. Будет ли этот ответ, и каким он будет, покажет ближайшее будущее.

В любом случае, мы живем в очень интересное время, время перемен в области управления данными. Займет ли новый мир место старого, привычного и для многих вполне удобного мира, или же два мира будут мирно существовать, покажет наступающее новое десятилетие.

Литература

[1] A Conversation with Michael Stonebraker and Margo Seltzer (http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=489), ACM Queue, Volume 5, Number 4, May/June 2007. Перевод: Беседа Марго Зельцер с Майклом Стоунбрейкером (http://citcity.ru/16453/).

[2] Michael Stonebraker, Uğur Çetintemel. «One Size Fits All»: An Idea Whose Time Has Come and Gone (http://www.cs.brown.edu/~ugur/fits_all.pdf). Перевод: Майкл Стоунбрейкер, Угур Кетинтемел. «Один размер пригоден для всех»: идея, время которой пришло и ушло (http://citforum.ru/database/articles/one_size_fits_all/).

[3] Michael Stonebraker, Chuck Bear, Uğur Çetintemel, Mitch Cherniack, Tingjian Ge, Nabil Hachem, Stavros Harizopoulos, John Lifter, Jennie Rogers, and Stan Zdonik. One Size Fits All? – Part 2: Benchmarking Results (http://nms.csail.mit.edu/~stavros/pubs/osfa.pdf). Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research (CIDR), January 7-10, 2007, Asilomar, California, USA. Перевод: Майкл Стоунбрейкер, Чак Беэ, Угур Кетинтемел, Мич Черняк, Тиньян Ге, Набил Хачем, Ставрос Харизопулос, Джон Лифтер, Дженни Роджерс, Стэн Здоник. Пригоден ли один размер для всех? Часть 2: результаты тестовых испытаний (http://citforum.ru/database/articles/one_size_fits_all_2/).

[4] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, Pat Helland. The End of an Architectural Era (It's Time for a Complete Rewrite) (http://citforum.ru/database/articles/end_of_arch_era/). Proceedings of VLDB, 2007, Vienna, Austria. Перевод: Майкл Стоунбрейкер, Сэмюэль Мэдден, Дэниэль

Абади, Ставрос Харизопулос, Набил Хачем, Пат Хеллэнд. Конец архитектурной эпохи, или Наступило время полностью переписывать системы управления данными (http://citforum.ru/database/articles/end_of_arch_era/).

[5] Сергей Кузнецов. Универсальность и специализация: время разбивать камни? (http://citforum.ru/database/articles/time_to_break_stones/).

[6] StreamBase (http://www.streambase.com/).

[7] Streaminsight (http://www.microsoft.com/sqlserver/2008/en/us/R2-complex-event.aspx)

[8] Vertica (http://www.vertica.com/).

[9] H-Store (http://db.cs.yale.edu/hstore/).

[10] SciDB (http://scidb.org/).

[11] Марк Ривкин. Тенденции развития универсальных коммерческих СУБД. (http://citforum.ru/database/articles/trends/).

[12] Future Directions in DBMS Research – The Laguna Beach Participants (http://www.icsi.berkeley.edu/pubs/techreports/tr-88-001.pdf). SIGMOD Record 18(1): 17-26, 1989. Пересказ на русском языке: Будущие направления исследований в области баз данных: десять лет спустя (http://www.citforum.ru/database/articles/future_01.shtml).

[13] Abraham Silberschatz, Michael Stonebraker, and Jeffrey D. Ullman. Database Systems: Achievements and Opportunities (http://infolab.stanford.edu/~hector/lagi.ps). CACM 34(10): 110-120, 1991.

[14] Abraham Silberschatz, Michael Stonebraker, Jeffrey D. Ullman. Database Research: Achievements and Opportunities Into the 21st Century (http://www.cs.rmit.edu.au/~zahirt/Teaching/oodb/Papers/view.ps). SIGMOD Record 25(1): 52-63 (1996). Перевод: Базы данных: достижения и перспективы на пороге 21-го столетия (http://citforum.ru/database/classics/nfs_report/).

[15] Avi Silberschatz, Stan Zdonik et al., Strategic Directions in Database Systems – Breaking Out of the Box (http://www.cs.rmit.edu.au/~zahirt/Teaching/oodb/Papers/view.ps). *ACM Computing Surveys*, Vol. 28, No. 4 (Dec 1996), 764-778. Перевод: Стратегические направления в системах баз данных (http://citforum.ru/database/classics/nsf_report2/).

[16] The Asilomar Report on Database Research (http://www.sigmod.org/record/issues/9812/asilomar.html). SIGMOD Record 27(4): 74-80, 1998. Перевод: Асиломарский отчет об исследованиях в области баз данных (http://citforum.ru/database/digest/asil_01.shtml).

[17] The Lowell Database Research Self-Assessment (http://research.microsoft.com/~Gray/Lowell/). CACM 48(5): 111-118, 2005. Пересказ на русском языке: Крупные проблемы и текущие задачи исследований в области баз данных (http://www.citforum.ru/database/articles/problems/).

[18] The Claremont Report on Database Research (http://db.cs.berkeley.edu/claremont/claremontreport08.pdf). Перевод: Клермонтский отчет об исследованиях в области баз данных (http://citforum.ru/database/articles/claremont_report/).

[19] Goetz Graefe. The Five-minute Rule: 20 Years Later and How Flash Memory Changes the Rules (http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=549),

ACM QUEUE, July/August 2008. Перевод: Гоц Грейф «Правило пяти минут двадцать лет спустя, и как флэш-память изменяет правила» (http://www.citforum.ru/database/articles/five_minute_rule/

[20] LINQ (http://msdn.microsoft.com/en-us/netframework/aa904594.aspx)

[21] Ruby on Rails (http://www.rubyonrails.ru/)

[22] salesforce.com (http://www.salesforce.com/)

[23] Amazon S3 (http://aws.amazon.com/s3)

[24] Google App Engine (http://code.google.com/appengine/)

[25] Amazon EC2 (http://aws.amazon.com/ec2)

[26] Amazon SimpleDB (http://www.amazon.com/SimpleDB-AWS-Service-Pricing/b?ie=UTF8&node=342335011)

[27] Microsoft SQL Server Data Services (http://www.microsoft.com/sql/dataservices/default.mspx)

[28] Google's Datastore (http://www.groovie.org/2008/04/13/google-datastore-and-the-shift-from-a-rdbms)

[29] Jeffrey Dean, Sanjay Ghemawat «MapReduce: Simplifed Data Processing on Large Clusters» (http://labs.google.com/papers/mapreduce-osdi04.pdf), Proceedings of the Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004

[30] MapReduce в Hadoop (http://hadoop.apache.org/mapreduce/)

[31] Database Column (http://databasecolumn.vertica.com/)

[32] Michael Stonebraker, David J. DeWitt. MapReduce: A major step backwards (http://databasecolumn.vertica.com/database-innovation/mapreduce-a-major-step-backwards/)

[33] Michael Stonebraker, David J. DeWitt. MapReduce II (http://databasecolumn.vertica.com/database-innovation/mapreduce-ii/)

[34] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Michael Stonebraker. A Comparison of Approaches to Large-Scale Data Analysis (http://cs-www.cs.yale.edu/homes/dna/papers/benchmarks-sigmod09.pdf). Proceedings of the 35th SIGMOD International Conference on Management of Data, 2009, Providence, Rhode Island, USA. Перевод: Эндрю Павло, Эрик Паулсон, Александр Разин, Дэниэль Абади, Дэвид Девитт, Сэмюэль Мэдден, Майкл Стоунбрейкер. Сравнение подходов к крупномасштабному анализу данных (http://citforum.ru/database/articles/mr_vs_dbms/)

[35] Greenplum (http://www.greenplum.com/)

[36] Eric Friedman, Peter Pawlowski, John Cieslewicz «SQL/MapReduce: A practical approach to selfdescribing, polymorphic, and parallelizable userdefined functions» (http://www.asterdata.com/resources/downloads/whitepapers/sqlmr.pdf), Proceeding of the VLDB '09 August 2428, 2009, Lyon, France

[37] MapReduce в Vertica (http://www.vertica.com/company/news/vertica-analytic-database-broadens-reach-with-flexstore)

[38] SQL Azur (http://www.microsoft.com/windowsazure/sqlazure/)

[39] SQL Server 2008 R2 Parallel Data Warehouse (http://www.microsoft.com/sqlserver/2008/en/us/parallel-data-warehouse.aspx)

[40] Daniela Florescu, Donald Kossmann. Rethinking Cost and Performance of Database Systems (http://www.sigmod.org/sigmod/record/issues/0903/p43.articles.florescu.pdf). SIGMOD Record, Vol. 38, No. 1, March 2009. Перевод: Переосмысление стоимости и производительности систем баз данных (http://citforum.ru/database/articles/rethinking/)

[41] 28msec (http://www.28msec.com/)

[42] Martin Kaufmann, Donald Kossmann. «Developing an Enterprise Web Application in XQuery» (http://www.28msec.com/download/enterprise_webapps.pdf)

[43] Jim Gray. (http://en.wikipedia.org/wiki/Jim_Gray_%28computer_scientist%29)

[44] Adam Jacobs. The Pathologies of Big Data (http://queue.acm.org/detail.cfm?id=1563874). ACM Queue, Vol. 7, Issue 6, July 2009. Перевод: Адам Якобса «Патологии больших данных» (http://citforum.ru/database/articles/pathology/)

[45] Сергей Кузнецов. О точности диагностики патологий (http://citforum.ru/database/articles/pathology_diagnostics/

[46] Tenbase (http://www.1010data.com/solutions.dw.tenbase.html)

[47] 1010data Inc. (http://www.1010data.com/)

[48] Jeffrey Cohen, Brian Dolan, Mark Dunlap, Joseph M. Hellerstein, Caleb Welton. MAD Skills: New Analysis Practices for Big Data (http://db.cs.berkeley.edu/jmh/papers/madskills-032009.pdf). Proceedings of the VLDB'09 Conference, Lyon, France, August 24-28, 2009). Пер.

[49] Перевод: Джеффри Коэн, Брайен Долэн, Марк Данлэп, Джозеф Хеллерстейн и Кейлэба Велтон «МОГучие способности: новые приемы анализа больших данных» (http://citforum.ru/database/articles/mad_skills/)

[50] Jim Gray, David T. Liu, Maria Nieto-Santisteban, Alex Szalay, David J. DeWitt, Gerd Heber. Scientific Data Management in the Coming Decade (http://www.sigmod.org/sigmod/record/issues/0512/p34-article-gray.pdf), SIGMOD Record, Vol. 34, No. 4, Dec. 2005. Перевод: Управление научными данными в следующем десятилетии (http://www.citforum.ru/database/articles/scientific_data/

[51] 50. Requirements for Science Data Bases and SciDB (http://www-db.cs.wisc.edu/cidr/cidr2009/Paper_26.pdf). Proceedings of the Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009

[52] A Demonstration of SciDB: A Science-Oriented DBMS (http://scidb.org/Documents/SciDB-VLDB09-paper.pdf). Proceedings of the VLDB ‘09, August 2428, 2009, Lyon, France