SCIDB: СУБД ДЛЯ НАУЧНЫХ ДАННЫХ

О. Бартунов, П. Велихов

Развитие технологий приемных устройств привело к необходимости хранения, обработки и анализа сверхбольших объемов научных данных. Современные компьютерные и информационные технологии не готовы для решения этих задач и требуются новые решения, ориентированные на работу с научными данными, доступные для научного сообщества и масштабируемые на сотни петабайт. В этой статье мы рассматриваем СУБД SciDB, как решение для большинства научных проектов.

Повсеместное распространение интернет, ускорение и унификация доступа к информации и т.п. привело к формулированию концепции кибероборщества (информационного общества) как реалистичного сценария постиндустриального общества - новой исторической фазы развития цивилизации, в которой главными продуктами производства являются информация и знание. Составной частью информационного общества является так называемая e-Science - синтез науки и информатики, наступающий когда роль информации и ее обработки в научных исследованиях становится превалирующей. Переход на е-стадию (информационную стадию) развития - это реальный процесс, затронувший на сегодняшний день ряд естественных наук, оперирующих огромными объемами информации: физика (в первую очередь исследование элементарных частиц и физика высоких энергий), науки о Земле, астрономия, биология, экономика, медицина. В этих науках происходит процесс лавинного поступления информации, в первую очередь связанный с успехами в технологии создания приемных устройств, появился даже термин «сенсорно-ориентированная наука». Ведущие издания (The Economist, Nature, CACM, Science) посвятили специальные номера, посвященные проблеме “Больших Данных” (BIG DATA). В современных крупных экспериментах анализ терабайтов и даже петабайтов научных данных становится повседневной задачей.

Мы описали вызовы системам управления данных о науке в журнале “Суперкопьютеры”, поэтому мы кратко резюмируем их в данной статье.

Специфика научных данных

Специфика научных данных состоит в необходимости "вечного" хранения “сырых” данных (raw data, - это данные, полученные непосредственно с приемника и не подвергшиеся никакой обработки). Сырые данные - это неизменяемые данные, любое их изменение приводит к появлению новых сырых данных (новой версии). Сырые данные необходимо обработать, чтобы получить, собственно научные данные, с которыми и работают ученые. С развитием науки и технологии, дистанция между сырыми и научными данными все время увеличивается, а процедура обработки сырых данных усложняется.

Современные научные эксперименты, такие как LHC (Большой Адронный Коллайдер), LSST (Большой телескоп для обзора неба), помимо сложных сенсоров, а значит и сложнейшей процедуры обработки, требуют значительных компьютерных ресурсов для передачи и хранения сырых данных.

Требование вечного хранения сырых данных и свободного доступа к ним определяется основным принципом научного исследования - воспроизводимость научных результатов. К этому надо добавить необходимость хранения и самой процедуры обработки сырых данных.

Исторически, СУБД предоставляли пользователям способы хранения и работы с данными, а сами данные готовились для загрузки отдельной процедурой. Очевидно, что подобное разделение не может поддерживать целостности и воспроизводимости научных результатов. Напрашивается естественный вывод - перенести сами сырые данные, их обработку (кукинг, cooking) и подготовку научных данных в СУБД, т.е. СУБД должна поддерживать полный цикл работы с данными: хранение сырых данных, обработку сырых данных, анализ научных данных, обмен данными.

Выбор СУБД для научных данных

В настоящее время насчитывается около сотни различных СУБД, от классических реляционных СУБД (Oracle, DB2, SQL Server, PostgreSQL, MySQL, Firebird, Ingres,…), которые обладают богатым набором возможностей, но их архитектура закладывалась во времена одного (не сетевого) большого и дорого компьютера с маленькой памятью и одноядерным процессором, до специализированных хранилищ, оптимизированных для решения определенных задач (Vertica, H-Store, StreamDB..). Посередине находятся СУБД, для которых самым важным является масштабирование и ограниченный набор возможностей. Эти СУБД ориентированы на современную многоядерную архитектуру дешевых серверов с большой памятью, организованных в кластера.

Какой же класс СУБД годится для науки ? Очевидно, что богатые возможности реляционных СУБД крайне интересны для науки, но также очевидно, что строгая целостность и изоляция данных (CI в ACID) не важны, так как данные в науке в основном WORM (Write Once Read Many) и вполне достаточна eventual

consistency (в конце-концов). Кроме того, реляционной модели объекты хранятся вне зависимости друг-от-друга, когда в "сенсорно-ориентированной" науки, данные хранятся в массивах, где существуют окрестности точек и расстояния между любыми двумя объектами, причем анализ данных в окрестностях точек очень важен для науки. В реляционной модели реализация массивов очень неэффективна.

В последнее время появилось много систем обеспечивающих высокую производительность запросов в параллельном режиме: Data Shards, Greenplum, Aster Data, HadoopDB и другие. Все они работают с реляционной моделью данных, которая плохо подходит под задачи обработки научных данных. Так как эти системы основаны на традиционных СУБД, они страдают низкой производительностью из-за большого количества ключей, которые надо получать для выполнения запросов, из-за записи данных в журнал и из-за управления страницами памяти. В системах, использующих MapReduce для достижения параллельности, пользователю самому приходится разрабатывать план запроса и план распределения данных по узлам на низкоуровневом языке программирования.

Если к требованию поддержки массивов добавить, специфические для науки требования, такие как поддержка версионности, происхождения, аннотирования данных, данных с ошибками итд, то приходим к выводу, что на сегодняшний момент нет СУБД, ориентированной на современную и будущую науку.

Сложившаяся ситуация в больших научных проектах была оценена ведущими учеными из разных наук,представителями коммерческих компаний и разработчиками в области СУБД (систем управления баз данных) на серии конференций XLDB, в результате чего возник проект SciDB под руководством профессора MIT Майка Стоунбрейкера, автора первой реляционной СУБД Ingres, СУБД Postgres, и множеством других успешных систем, и его коллег из крупнейших университетов США. Проект был основан в 2008 году, и помимо американских ученых, в нем задействованы Российские ученые и разработчики из НИИСИ РАН и ГАИШ МГУ.

Основная цель проекта - разработка в кратчайшие сроки СУБД для нужд больших научных и промышленных проектов, в которых требуется анализ сверхбольших объемов данных (сотни и тысячи петабайт), масштабируемая на тысячи серверов.

SciDB: СУБД для больших объемов научных данных

Система SciDB разрабатывается в первую очередь исходя из требований больших научных проектов и имеет ряд принципиальных отличий от существующих СУБД. SciDB разрабатывается как система для хранения и анализа сырых и производных научных данных. Некоторые основные функции традиционных баз данных не поддерживаются в SciDB, позволяя системе более эффективно обрабатывать аналитические запросы. Например, так как исходные данные фактически не обновляются, в SciDB не предусмотрена эффективная поддержка больших объемов транзакций, что позволяет избежать серьезных накладных расходов. Наконец, SciDB – проект с открытым исходным кодом и бесплатной лицензией на использование, что отвечает требованиям большинства заказчиков. Открытый код позволяет экономить средства заказчиков на масштабные внедрения системы, а открытый процесс разработки обеспечивает высокое качество технических решений. Кроме того, открытость СУБД обеспечивает технологическую независимость и возможность обмена данными между разными научными коллективами. SciDB - это массивно параллельная СУБД, готовая к анализу большого объема научных данных, но также работает на персональных компьютерах, предоставляя пользователям единую среду обработки и анализа данных. Таким образом пользователь может отлаживать свои алгоритмы обработки на небольших выборках данных, и те же самые запросы и функции запускать на больших, параллельных системах.

Кроме привычных функций систем управления базами данных, в SciDB присутствуют новые механизмы работы с данными, специально разработанные для анализа научных данных. Модель данных SciDB представляет из себя многомерные вложенные массивы, таким образом ученым не надо моделировать свои данные как таблицы записей, что в свою очередь ведет к более простой формулировке аналитических запросов и на порядки увеличивает производительность системы. Так как в SciDB будут хранится данные полученные с приборов, SciDB поддерживает погрешность измерений на уровне модели данных и языка запросов.

SciDB – это массивно параллельная СУБД с архитектурой shared nothing, то есть каждый узел SciDB работает только с локальными данными и памятью. Архитектура SciDB приведена ниже:

Сырые данные, поступающие на кластер раскидываются системой на подмассивы с перехлестом, размер массивов и перехлеста задается пользователем. Кластер управляется координатором для каждого запроса, то есть в системе обычно работает несколько координаторов. Каждый координатор компилирует запросы в планы и рассылает их рабочим узлам, которые на основе лежащих в них данных выполняют запросы. Если требуются дополнительные данные на рабочих узлах, то в план запроса вставляется оператор Scatter/Gather, который рассылает дополнительные данные.

Рассмотрим пример из астрономии, где в SciDB хранятся изображения с ПЗС матрицы. Каждый узел хранит какое-то подмножество данных, причем на логическом уровне данные представлены как массив кортежей, <x, y, l, err> в нашем примере (x,y – это координаты точки на CCD, l – значение сигнала, err – ошибка измерения). На физическом уровне массивы разных атрибутов хранятся отдельно. Такая модель получила название вертикальной и часто используется в современных СУБД. Основные преимущества такой модели в том, что данные лучше сжимаются, так как сжимаются данные одного типа, и при запросах,

выбираются только те атрибуты, которые необходимы для вычисления запроса. Пользователь СУБД указывает какой алгоритм сжатия использовать для каждого атрибута, следовательно разные атрибуты из одного массива могут сжиматься разными функциями. Также, на физическом уровне массивы атрибутов хранятся в чанках, то есть больших страницах, которые тоже включают в себя перехлест. Таким образом, единица обработки данных в SciDB является чанк с перехлестом. Данная архитектура хранения показана ниже:

Исполнение запросов в SciDB конвеерное, многим напоминающее конвеерное исполнение запросов в реляционных базах. Только единицей обработки данных в SciDB является чанк. Так как мы имеем дело с массивами, конвеерное исполнение происходит по функции getChunk(position), то есть запрашивается чанк по позиции из верх-стоящего оператора. Рассмотрим пример оператора Subarray, который вырезает кусок массива из исходного массива. У оператора есть две точки, нижняя и верхняя, по которым происходит вырезка. Получая запрос getChunk(position), оператор находит искомый чанк во входе, посылая запрос getChunk(position + low), где low - это нижняя точка вырезки. Получив чанк, оператор смотрит, лежит ли чанк полностью внутри двух точек, если так, то чанк посылается наверх без изменений и распаковки. Иначе, чанк распаковывается и из него вырезаются ненужные точки. Другие операторы SciDB работают по похожей схеме, что позволило создать конвеерный вычислитель запросов для SciDB.

Как известно, научные запросы к массивам данных часто требуют локальных вычислений, то есть вычислений в небольшой окрестности каждой точки. Например в анализе изображений часто используют фильтры для сглаживания изображений. SciDB поддерживает запросы, включающие в себя анализ окрестности точек используя чанки с перехлестом. Если перехлеста данных достаточно для запроса, то SciDB параллельно выполняет запрос на всех узлах. Если перехлест недостаточен, то в план вставляется оператор Scatter/Gather, который увеличивает перехлест.

Также, модель хранения данных в SciDB предусматривает репликацию для обеспечения отказоустойчивости системы. Каждый чанк массивов, хранящихся в SciDB, реплицируется на других узлах, а при работе SciDB координатор проверяет задействованные рабочие узлы, чтобы вовремя понять, что какой-то узел вышел из строя. В этом случае, для обработки запроса используются резервные данные, лежащие на других узлах.

При разработке SciDB учитывалось, что научные данные меняются редко, и что необходимо хранить и иметь доступ ко всем версиям данных. Однако, в отличие от коммерческих СУБД, не требуется поддержка большого потока транзакций. Поэтому при разработки системы удалось избежать использование большого количества синхронизационных механизмов, которые присутствуют в традиционных СУБД. Это серьезно ускоряет работу системы, особенно в режиме параллельного доступа к данным, где традиционным системам приходится использовать распределенные менеджеры замков. В SciDB нету изменений данных - пользователь может только создавать новые версии данных. Каждое новое изменение влечет за собой создание новой версии. Если изменение достаточно большое, то создаются новые чанки, а у незатронутых чанков только изменяются мета-данные. Этот метод называется COW (Copy On Write) и эффективен при больших изменениях в данных. Если же меняется небольшое количество данных чанка, то работает схема дельт, то есть последняя версия хранится в чанке, а предыдущая доступна применением небольшой дельты к чанку. Дельты храняться вместе с чанками, и пользователь задает процент резервации для хранения дельт. Создание новой версии данных является атомарной процедурой, и запрос всегда возвращает корректную версию данных. Таким образом SciDB, хоть и поддерживает изменения данных, но делает это эффективно.

Наличие декларативного языка запросов к массивам обеспечивает прозрачный доступ к терабайтам многомерных данных. В SciDB реализован язык AQL (Array Query Language), во многом похожий на язык SQL. В AQL присутствует такая же конструкция SELECT FROM WHERE, только язык оперирует с массивами, а не с множествами. Язык запросов AQL позволяет формулировать запросы, анализирующие окрестности точек. Разбиение данных по чанкам с перехлестами позволяет выполнять такие запросы параллельно без коммуникации вычислительных узлов. Один из основных операторов AQL – это REGRID, который во многом похож на популярный MapReduce. REGRID создает новый массив данных на основе исходного, пользуясь двумя функциями - доменной и агрегатной. Доменная функция выбирает подмножество массива для подсчета, а агрегатная функция подсчитывает значение элемента выходного массива. Используя REGRID гораздо проще выбирать точки с их окрестностями, а модель хранения чанков с перехлестом позволяет вычислять выходной массив параллельно на всех узлах без коммуникаций. Рассмотрим пример использования оператора REGRID для сглаживания исходных данных гауссовским фильтром:

```
SELECT l FROM CCD AS C
REGRID (
SELECT l FROM CCD AS C1 WHERE C1.i BETWEEN C.i-20 AND C.i+20
    AND C1.j BETWEEN C.j-20 and C.j+20,
SUM( C1.l * a*e^( (-i-b1)^2/(2*c1^2) + (-j-b2)^2/(2*c2^2)))
```

В этом примере первый параметр в операторе REGRID выбирает окрестность точек 40x40 вокруг

входной точки массива, а второй оператор считает гауссовскую функцию на основе выбранных точек.

В реляционных базах такие запросы невозможны, даже простая выборка данных, когда пользователю надо выбрать куб данных - то, что делает доменная функция - уже сложная задача для реляционных баз. Для этого реляционным базам требуется многомерный индекс или придется пробежаться по всем данным таблицы, в то время как в SciDB это встроенная возможность.

В системе MapReduce такого рода запросы в принципе возможны, можно использовать функцию map чтобы выбирать окрестности точек и функцию reduce для подсчета агрегатов. Но при этом усложняется работа пользователя и теряется эффективность: написать фунцию map, которая эквивалентна доменной функции regrida и выбирает чанки с перехлестом бывает сложно и map разбрасывает данные, совпадающие по ключам, по узлам для параллельного слияния. А regrid использует исходное разбиение данных и в большом количестве случаев не требует перераспределения данных по узлам. Таким образом regrid является более удобной и одновременно более эффективной версией MapReduce для работы с научными данными.

SS-DB - Бенчмарк для научных данных и СУБД

В 2010 году часть участников проекта SciDB разработала бенчмарк для научных данных и СУБД. Бенчмарк включается в себя четыре стадии работы с данными:

1. Усвоение сырых данных
2. Операции над сырыми данными
3. Получение научных данных (кукинг)
4. Работа с научными данными

Бенчмарк частично основан на требованиях проекта LSST, но тем не менее является достаточно общим для всех сенсорных наук. Запросы в SS-DB запускаются против массивов с одним измерением (вершины полигонов), двумя измерениями (изображения с сенсоров/научными данными) и тремя измерениями (траектории в пространстве и времени). Бенчмарк произведен с использованием системы SciDB и MySQL, последняя система выбрана так как она используется в большом количестве научных проектов, а так же является достаточно легковесной системой с хорошими показателями скорости выполнения запросов.

Сырые данные в SS-DB состоят из двумерных изображений с 7500 пикселей в каждом измерении, то есть 56250000 пикселей. В каждом массиве 11 атрибутов, 32-х битных целых чисел. 5 атрибутов выбраны из настоящих астрономических данных, где некоторые регионы представляют из себя “интересные” данные, а остаток - фоновые данные. Бенчмарк включает в себя 400 таких массивов, “снятых” в регулярные промежутки времени, общий объем бенчмарки - 1Тб данных. Массивы организованы по циклам, всего в бенчмарке 20 циклов по 20 массивов в каждом, что моделирует периодичность поступления сырых данных. Сверху координат массивов построена глобальная координатная система, от 0 до 10^8. Естественно, изображения не заполняют всю систему координат, вместо этого 80% изображений формируют плотное подмножество данных в центре координатной системы, а остальные 20% - разреженные данные в остальном диапазоне значений.

Кукинг данных работает следующим образом - функция детекции объектов пробегает все изображения и генерит примерно 4 Мб наблюдений для каждого массива, то есть 1.6 Гб научных данных. Каждое наблюдение состоит из: (x,y) центра наблюдения в глобальной координатной системе; границы наблюдения представляющую из себя полигон, в среднем полигон имеет 12 вершин, если наблюдение содержит в себе больше 12 вершин, то такое наблюдение отфильтровывается, чтобы не создавать выбросов; два значения с плавающей точкой, представляющие из себя значения из доменной области. Наблюдения в свою очередь объединяются в группы, так как одно и то же наблюдение может оказаться в разных изображениях из-за периодичности получения данных с сенсоров, из-за движения объекта или смещения сенсора. Группировка производится с учетом максимальной скорости объектов, и один и тот же объект может попасть в несколько групп. Наблюдения не попавшие в группу, создают свою собственную группу.

Кроме детекции объектов и группировки, SS-DB определяет 9 запросов, которые должны выполняться системами:

1. **Q1-aggregation** - посчитать среднее значение одного из атрибутов по вырезке [X,Y] - [U,V] и 20 изображениям каждого цикла - эта задача похожа на подсчет фонового шума сенсора.
2. **Q2-recooking** - заново провести кукинг данных для одного изображения.
3. **Q3-regridding** - ужать изображение 10:3 с помощью функции интерполяции
4. **Q4-aggregation** - посчитать среднее значение наблюдений с центрами в определенном регионе размером [X,Y] - [U,V] и во всех циклах
5. **Q5-polygons** - выбрать все наблюдения, чьи полигоны пересекаются с куском массива заданного [X-Y] - [U,V]
6. **Q6-density** - сгруппировать данные наблюдений из регионов [X,Y] - [U,V] и выбрать группы включающие в себя D или больше элементов.
7. **Q7-centroid** - найти все группы наблюдений чей центр попадает в регион [X,Y] - [U,V]
8. **Q8-trajectory** - траектория - это последовательность центров наблюдений в группе наблюдений. Для всех траекторий, которые пересекают регион [X,Y] - [U,V], выбрать из сырых

данных квадрат, отцентрованный по центру наблюдения, для всех изображений пересекающих регион.

9. **Q9-trajectory** - Альтернативное определение траектории - это последовательность полигонов, определяющих границу группы. Для всех групп, найти группу, чья траектория пересекается с регионом [X,Y] - [U,V], и выбрвть из сырых данных квадрат, отцентрованный по центру наблюдения, для всех изображений пересекающих регион.

Бенчмарки были осуществлены на кластере из 10 машин с ОС Ubuntu Karmic, с 2 Гб операционной памяти на узле и двух-коровым процессором Intel Xeon 3.2 Ghz, объединенными гигабитным эзернетом. Результаты бенчмарка приведены в таблице (время в минутах):

DBMS	Load	Obsv	Group	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
MySQL	770	200	90	54	44	161	50	32	51	52	395	395
SciDB	67	1.9	15	3.6	0.07	1.7	0.015	0.017	0.02	0.11	2.2	2.3

Как показывают результаты, SciDB в среднем на два порядка быстрее справляется с запросами, чем MySQL, что подтверждает тезис о востребованности специальной СУБД для научных данных. Мы допускаем что некоторые СУБД будут иметь лучшие показатели на бенчмарке, но не на порядки быстрее, так как запросы для MySQL кодировались на C++ и с помощью embedded SQL и отлаживались экспертами по тюнингу MySQL.

Полноценная поддержка полного цикла работы с научными данными

Как упоминалось раньше, из-за недостатков существующих СУБД, большинство научных проектов, в которых встает задача анализа больших объемов данных, осуществляют обработку и анализ исходных данных вне системы управления базами данных. SciDB решает эту проблему, обеспечивая эффективное и удобное хранилище исходных данных и широкий набор инструментов для обработки и анализа данных. Версионное хранилище и учет всех преобразований данных позволяет пользователям SciDB получить точную информацию о версиях данных и о всех вычислениях, произведенных над исходными данными. Это позволяет эффективно устранять ошибки в алгоритмах переработки данных, отслеживать процесс переработки исходных данных при получении подозрительных результатов, и в точности повторять вычисления над исходными данными. При этом SciDB работает без каких-либо ограничений, как на суперкомпьютерном кластере, так и на персональном компьютере, что позволит ученым работать в одной и той же среде со своими данными. Система SciDB также отслеживает происхождение данных. При обработке сырых данных и полученных из них объектов, SciDB запоминает запросы, с помощью которых были получены те или иные результаты. По требованию пользователя СУБД может проиграть процесс получения результатов из сырых данных или выдать выборку базы данных, которая требуется для получения результата. После переработки исходных данных, SciDB позволяет делиться полученными результатами, осуществлять выборки и выполнять аналитические запросы широкому кругу коллег, при этом соблюдая политику доступа как к данным, так и полученным результатам.

Таким образом, SciDB поддерживает полный цикл обработки и анализа данных, начиная от хранения сырых данных, и заканчивая анализом полученных результатов. При этом все результаты полученные в СУБД повторяемы и могут быть воспроизведены пользователем системы.

К настоящему времени выпущена версия 0.75, в которой еще много ограничений, однако тестирование системы на научных данных и типичных запросах уже показало лучшую производительность по сравнению с реляционными СУБД. Полноценная версия 1.0 готовится к выпуску в июне 2011 года.

ЛИТЕРАТУРА:

1. «Data, data everywhere», The Economist 25.02.2010
2. «Specials: Big Data», Nature, 03.09.2008
3. «Surviving the Data Deluge», CACM, 08.12, Vol 51, No 12
4. Science, 11.02.2011
5. О.Бартунов, П.Велихов, «Научные вызовы технологиям СУБД», Суперкомпьютеры, 5(15), 2011
6. О.Бартунов и др., «SciDB: Новая СУБД для больших объемов научный данных», Суперкомпьютеры, 5(15), 2011
7. XLDB: http://www-conf.slac.stanford.edu/xldb/
8. SciDB: http://www.scidb.org
9. MapReduce: http://en.wikipedia.org/wiki/MapReduce
10. M. Stonebraker et al., «SS-DB: A Standard DBMS Science Benchmark», XLDB 2010
11. M. Stonebraker et al., «A Demonstration of SciDB: A Science Oriented DBMS», VLDB 2009