



# Программирование приложений массивно- параллельных баз данных

Сергей Кузнецов, ИСП РАН

# Массивно-параллельные СУБД и архитектура shared-nothing

- ▶ Принято считать, что горизонтальную масштабируемость массивно-параллельных СУБД может обеспечить только архитектура shared-nothing
  - ▶ Здесь мы говорим только об аналитических базах данных
- ▶ Экземпляры СУБД в узлах системы не используют общих ресурсов и обрабатывают только хранимые в них данные
- ▶ Данные разделяются между узлами
- ▶ Общение между узлами происходит путем обмена сообщениями

# Что умеют делать массивно-параллельные СУБД

- ▶ Компилятор SQL знает о разделении данных между узлами системы и компилирует запрос таким образом, чтобы большая его часть выполнялась локально в узлах
- ▶ Минимизируются число обменов между узлами и объем передаваемых данных
- ▶ В результате при правильном (соответствующем рабочей нагрузке) разделении данных запросы выполняются параллельно на всех узлах системы
- ▶ Горизонтальная масштабируемость обеспечивается сравнительно легко
  - ▶ Требуется перераспределение данных

# В чем состояла проблема массивно-параллельных СУБД

- ▶ Еще одним важным принципом современных СУБД является перенос вычислений ближе к хранимым данным
- ▶ Для реализации пользовательских приложений на стороне серверов баз данных поддерживаются механизмы определяемых пользователем процедур, функций и методов
- ▶ Можно использовать разнообразные языки программирования (в частности, C/C++ и Java)
- ▶ Но программы, естественно, получаются последовательными
- ▶ Это сводит на нет преимущества массивно-параллельных СУБД и потенциальные возможности горизонтального масштабирования

# Почему проблема не решается с помощью традиционных подходов параллельного программирования

- ▶ Теоретически можно было бы внедрить возможности серверного параллельного программирования на основе MPI
- ▶ Но в данном случае это не получается
- ▶ MPI - это средство профессионального параллельного программирования
- ▶ Серверные приложения баз данных пишут в основном программисты-аналитики, которым гораздо более близки методы статистики, а не средства передачи сообщений и явной синхронизации
- ▶ Требовалось нечто менее универсальное, более простое и близкое по духу аналитикам
- ▶ Решение пришло из мира распределенных систем

# Выручил map/reduce

- ▶ Десять лет назад в Google возникла идея простой схемы реализации аналитической распределенной обработки данных map/reduce
  - ▶ Суть парадигмы считаю известной
- ▶ В сообществе баз данных поначалу резко отрицательно отнеслись к попытке заменить технологию массивно-параллельных баз данных фактически явным программированием средств обработки данных
- ▶ Однако пять лет спустя разработчики параллельных СУБД осознали, что на самом деле получили подарок
- ▶ Map/reduce реально обеспечивает возможность параллельного серверного программирования баз данных

# Почему map/reduce подходит

- ▶ Эта парадигма проста и близка аналитикам
  - ▶ map - это расширенный аналог привычной аналитикам операции group by
  - ▶ reduce - расширенный аналог агрегатной функции
- ▶ Параллельность программирования не видна пользователям
- ▶ Они по-прежнему заботятся о создании новых аналитических инструментов
- ▶ При этом массивно-параллельные СУБД становятся в полном объеме массивно-параллельными
- ▶ Возможности горизонтального масштабирования становятся более реальными

# Не нужен ли map/reduce в мире HPC?

- ▶ Видно, что в результате map/reduce занял в мире баз данных место, для которого вовсе не предназначался
- ▶ В общем-то, это вычисления под управлением данных в стиле SIMD
- ▶ Нет ли в мире HPC приложений, для которых эта парадигма также окажется близкой?
- ▶ Я имею в виду не конкретную реализацию (например, Hadoop), а саму идею
- ▶ Map/reduce не идеален, но эта парадигма близка народу



# Массивно-мультитредные архитектуры

- ▶ Несколько лет тому назад Леонид Константинович Эйсымонт увлек меня идеей массивно-мультитредных компьютеров
- ▶ Вдохновляет идея машины без кэша, которая работает со скоростью кэша без требования соблюдения принципа локальности за счет наличия практически неограниченного числа аппаратно поддерживаемых потоков
- ▶ Потенциал громаден, но непонятно (а) как программировать для такого компьютера; (б) что такое операционная система для такого компьютера (не Linux же!) и нужна ли она вообще и т.д.
- ▶ И мне пришло в голову, что массивно-мультитредные программы потенциально мог бы генерировать компилятор SQL

# Почему компилятор SQL может (?) это делать

- ▶ SQL - декларативный язык
- ▶ Компилятор SQL строит процедурные планы выполнения запросов
- ▶ Оператор SELECT состоит из практически неограниченного числа блоков, типов которых очень немного
- ▶ Основное время работы оператора выборки уходит на фильтрацию данных по условию
- ▶ Технически идея генерации массивно-мультитредной программы не проработана, реализация может оказаться очень сложной, но она кажется вполне возможной
- ▶ Естественно, компилятор должен работать на какой-то внешней машине

# Что можно получить

- ▶ СУБД, хранящую данные в основной памяти и работающую со скоростью кэша
- ▶ Это действительно система реального времени
- ▶ Не уверен, что сегодня СУБД с такими возможностями были бы востребованы, но потенциально могли бы появиться новые классы приложений баз данных
- ▶ Если же добавить к этой картине потенциальную возможность использования энергонезависимой памяти прямого доступа, то можно было бы получить полноценную СУБД с долговременно хранимыми данными

# Проблемы

- ▶ Конечно, все это мечты
- ▶ Предположим, что они сбудутся
- ▶ Остаются проблемы, подходов к решению которых я не вижу
- ▶ Запросы будут выполняться очень быстро, компилятор запросов не должен работать дольше, чем выполняется сам запрос
- ▶ Как этого добиться?
- ▶ Как писать «серверные» пользовательские приложения в такой архитектуре?
- ▶ Нужен еще один «map/reduce», представляющий собой непонятно что
- ▶ Та же проблема: как программировать для массивно-мультитредной архитектуры

# Давайте жить дружно!

- ▶ Все это ужасно интересно
- ▶ Мне кажется, что на пути к новым высокопроизводительным системам обработки данных нужна совместная работа сообществ баз данных и высокопроизводительных вычислений

Спасибо за внимание!