НАУЧНЫЙ ЖУРНАЛ
НАУКА И МИРОВОЗЗРЕНИЕ

ОСОБЕННОСТИ ХРАНЕНИЯ БОЛЬШИХ ДАННЫХ

Овездурдыева Ирина Курбангельдыевна

Старший преподаватель Туркменского государственного университета имени Махтумкули г. Ашхабад Туркменистан

Чарыева Мяхри Гурбандурдыевна

Преподаватель института Телекоммуникаций и информатики Туркменистана г. Ашхабад Туркменистан

Аннотация. Большие данные поставили перед традиционными системами хранения и обработки новые сложные задачи. В данной статье анализируются возможные способы их решения, ограничения, которые не позволяют сделать это эффективно, а также приводится обзор трех современных подходов к работе с большими данными: NoSQL, MapReduce и обработка потоков событий в реальном времени.

Ключевые слова: большие данные; реляционная модель; nosql; mapreduce; hadoop; s4; storm

Введение

Мы живем в информационный век. Нелегко измерить общий объем электронных данных, но по оценкам IDC размер «цифровой вселенной» в 2006 г. составлял 0.18 зеттабайт, а к 2011 г. должен был достигнуть 1.8 зеттабайт, продемонстрировав десятикратный рост за 5 лет! Вот только несколько примеров источников таких объемов

1. Нью-Йоркская фондовая биржа генерирует около терабайта данных в день.

2. Объем хранилища социальной сети Facebook каждый день увеличивается на 500 терабайт.

3. Проект Internet Archive уже хранит 2 петабайта данных и прирастает 20 терабайтами в месяц.

4. Эксперименты на Большом адронном коллайдере могут генерировать около петабайта данных в секунду!

Стремительно растущий объем информации ставит перед нами новые сложные задачи по организации ее хранения и обработки. В статье мы постараемся проанализировать то, как изменился традиционный подход к организации данных, а также проанализируем новые инструменты работы с большими данными.

Что такое «большие данные»

Итак, данных очень много. Но что такое «много»? Где тот порог, преодолев который, данные становятся «большими»?

Часто используется характеристика, данная исследовательской компанией Gartner: ««Большие данные» характеризуются объемом, разнообразием и скоростью, с которой структурированные и неструктурированные данные поступают по сетям передачи в процессоры и хранилища, наряду с процессами преобразования этих данных в ценную для бизнеса информацию»

Как видно из этого определения, большие данные имеют четыре основные характеристики: объем, разнообразие, скорость и ценность. Рассмотрим их подробнее:

1. Объем. Нарастающее количество данных, создаваемых как людьми, так и машинами, предъявляет к IT инфраструктуре новые требования в отношении хранения, обработки и предоставления доступа.

2. Разнообразие. Данные содержат разнообразную информацию, представленную разными структурами. Со всем этим, от логов доступа к веб-серверу до операций по кредитным картам, от результатов научных экспериментов до фотографий и видео, необходимо уметь работать.

3. Скорость. Важно осознавать, что под скоростью понимается не только скорость, с которой данные поступают в хранилище, но и скорость с которой важная информация из этих данных извлекается.

4. Ценность. Большие объемы данных — это ценный ресурс. Но еще ценнее он становится, если позволяет отвечать на насущные вопросы или вопросы, которые могут возникнуть в будущем.

Так сложилось, что инструменты, существовавшие до недавнего времени, оказались не способны справиться с большими объемами. О проблемах, пришедших с эпохой больших данных, способах их преодоления и новых инструментах поговорим в следующих разделах.

До определенного момента, практически единственным ответом на вопрос «как хранить и обрабатывать данные?» являлась какая-нибудь реляционная СУБД. Но с увеличением объемов появились проблемы, с которыми классическая реляционная архитектура не справлялась, поэтому инженерам пришлось придумывать новые решения. Попробуем представить те шаги, которые можно предпринять, если СУБД прекращает справляться с объемом выполняемых операций:

1. Естественным первым шагом является попробовать наименее затратные способы. Самый простой, при наличии финансов, способ — это ничего не делать, а просто купить более мощное оборудование (вертикальное масштабирование). Однако бесконечно мощного сервера не существует, а значит вертикальный рост конечен.

2. Более затратный способ — это оптимизировать запросы, проанализировав планы их исполнения, и создать дополнительные индексы. Такой метод может принести временное облегчение, но дополнительные индексы порождают дополнительные операции, а с ростом объемов обрабатываемых данных эти дополнительные операции приводят к деградации.

3. Следующим шагом может быть внедрение кэша на чтение. При правильной организации такого решения, мы можем избавить СУБД от существенной части операций чтения, но жертвуем строгой консистентностью данных. К тому же, этот подход приводит к усложнению клиентского ПО.

4. Выстраивание операций вставки/обновления в очередь — неплохое решение, но размер очереди ограничен. К тому же, для обеспечения строгой консистентности, нам придется организовать персистентность самой очереди, а это непростая задача.

5. Наконец, когда все прочие способы перестают работать, наступает момент пересмотреть способ организации самих данных. В первую очередь — произвести денормализацию схемы, чтобы уменьшить число нелокальных обращений.

6. Ну а когда и это не работает, то остается только масштабировать горизонтально, т.е. разносить вычисления на разные узлы. Здесь приходится окончательно попрощаться с нормализацией и внешними ключами, к тому же нужно ответить на вопросы «по каким признакам распределять новые кортежи по узлам?» и «как произвести миграцию существующей схемы?».

Подводя итоги, можно заключить, что попытки приспособить реляционную СУБД к работе с большими данными приводят к следующему:

1. Отказу от строгой консистентности.

2. Уходу от нормализации и внедрение избыточности.

3. Потере выразительности языка SQL и необходимости моделировать часть его функций программно.

4. Существенному обеспечению. усложнению клиентского программного

5. Сложности поддержания работоспособности и отказоустойчивости получившегося решения.

Необходимо, правда, отметить, что производители реляционных СУБД осознают все эти проблемы и уже начали предлагать масштабируемые кластерные решения. Однако стоимость внедрения и сопровождения подобных решений зачастую не окупается.

Взглянув на выводы из предыдущего раздела, в голове сразу рождается довольно очевидная мысль — а почему бы не спроектировать архитектуру, способную адаптироваться к возрастающим объемам данных и эффективно их обрабатывать? Подобные мысли привели к появлению движения NoSQL. Сразу хочется обратить внимание, что NoSQL не подразумевает бездумного отказа от всех принципов реляционной модели. Более того, термин «NoSQL» впервые был использован в 1998 году для описания реляционной базы данных, не использовавшей SQL [3]. Просто теоретики и практики данного подхода справедливо утверждают, что при выборе инструментария необходимо отталкиваться от задачи, а реляционные СУБД подходят не всегда, особенно в эпоху больших данных. Популярность NoSQL стал набирать в 2009 г, в связи с появлением большого количества веб-стартапов, для которых важнейшей задачей является поддержание постоянной высокой пропускной способности хранилища при неограниченном увеличении объема данных. Рассмотрим основные особенности NoSQL подхода:

1. Исключение излишнего усложнения. Реляционные базы данных выполняют огромное количество различных функций и обеспечивают строгую консистентность данных. Однако для многих приложений подобный набор функций, а также удовлетворение требованиям ACID являются излишними.

2. Высокая пропускная способность. Многие NoSQL решения обеспечивают гораздо более высокую пропускную способность данных нежели традиционные СУБД.

Например, колоночное хранилище Hypertable, реализующее подход Google Bigtable, позволяет поисковому движку Zvent сохранять около миллиарда записей в день. В качестве другого примера можно привести саму Bigtable, способную обработать 20 петабайт информации в день

3. Неограниченное горизонтальное масштабирование. В противовес реляционным СУБД, NoSQL решения проектируются для неограниченного горизонтального масштабирования. При этом добавление и удаление узлов в кластере никак не сказывается на работоспособности системы. Дополнительным преимуществом подобной архитектуры является то, что NoSQL кластер может быть развернут на обычном аппаратном обеспечении, существенно снижая стоимость всей системы.

4. Консистентность в жертву производительности. При описании подхода NoSQL нельзя не упомянуть теорему CAP. Следуя этой теореме, многие NoSQL базы данных реализуют доступность данных (availability) и устойчивость к разделению (partition tolerance), жертвуя консистентностью в угоду высокой производительности. И действительно, для многих классов приложений строгая консистентность данных — это то, от чего вполне можно отказаться.