Использование нереляционных БД при построении масштабируемых веб-приложений

Ilya Bakulin

SMS Traffic

LVEE 2010

Болезни роста

- Недостаточная пропускная способность сети
- Недостаточные мощности Application-серверов либо серверов БД
- Неоптимальная архитектура приложения

Сервера БД — узкое место

Что с этим делать?

- Оптимизировать структуру хранения данных, таблицы для агрегированных данных
- Денормализация
- Удалять индексы, поддерживающие ссылочную целостность
- Partitioning/sharding
- Смена движка БД

CAP theorem

Для тех, кто видит сны

Consistency, **A**vailability, **P**artition tolerance
Невозможно построить продукт, обладающий хорошими характеристиками по более чем двум из перечисленных признаков, и при этом не дающий ощутимых задержек при обращении.

- Реляционные БД обычно обеспечивают **CA**
- NoSQL-решения, в т.ч. Cassandra, тяготеют к **AP**

BASE: **B**asically **A**vailable **S**oft-state **E**ventually consistent — основная характеристика подобных решений, в отличие от **ACID** в RDMBS.

CAP theorem

Для тех, кто видит сны

Consistency, **A**vailability, **P**artition tolerance
Невозможно построить продукт, обладающий хорошими характеристиками по более чем двум из перечисленных признаков, и при этом не дающий ощутимых задержек при обращении.

- Реляционные БД обычно обеспечивают **CA**
- NoSQL-решения, в т.ч. Cassandra, тяготеют к **AP**

BASE: **B**asically **A**vailable **S**oft-state **E**ventually consistent — основная характеристика подобных решений, в отличие от **ACID** в RDMBS.

CAP theorem

Для тех, кто видит сны

Consistency, **A**vailability, **P**artition tolerance
Невозможно построить продукт, обладающий хорошими характеристиками по более чем двум из перечисленных признаков, и при этом не дающий ощутимых задержек при обращении.

- Реляционные БД обычно обеспечивают **CA**
- NoSQL-решения, в т.ч. Cassandra, тяготеют к **AP**

BASE: **B**asically **A**vailable **S**oft-state **E**ventually consistent — основная характеристика подобных решений, в отличие от **ACID** в RDMBS.

NoSQL как концепция

- Простые структуры хранения данных
- Время доступа к данным постоянно и не зависит от объёма данных, хранимых системой
- What else?

Немного истории

- Разработана в Facebook для решения задачи поиска по инбоксу пользователей
- Код открыт в 2008-м году (Java, GPLv2)
- В настоящее время входит в число первичных проектов Apache Software Foundation

Где используется

- Facebook :-)
- Reddit
- Digg
- Twitter

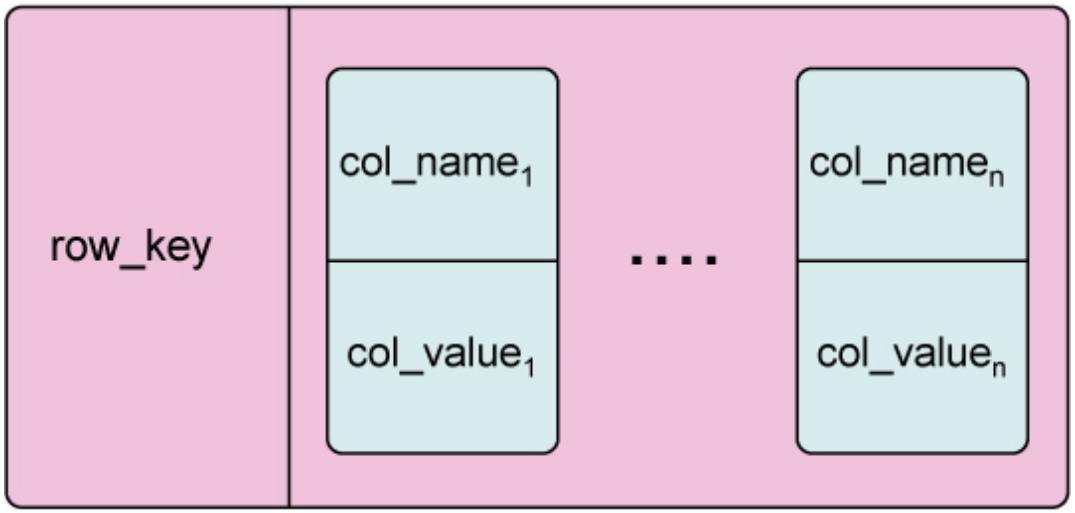
Модель данных
Унаследована от Google BigTable

- Интересная модель данных
- Гибкое конфигурирование репликации
- Реализация протокола доступа (Thrift) есть для многих ЯП

Модель данных

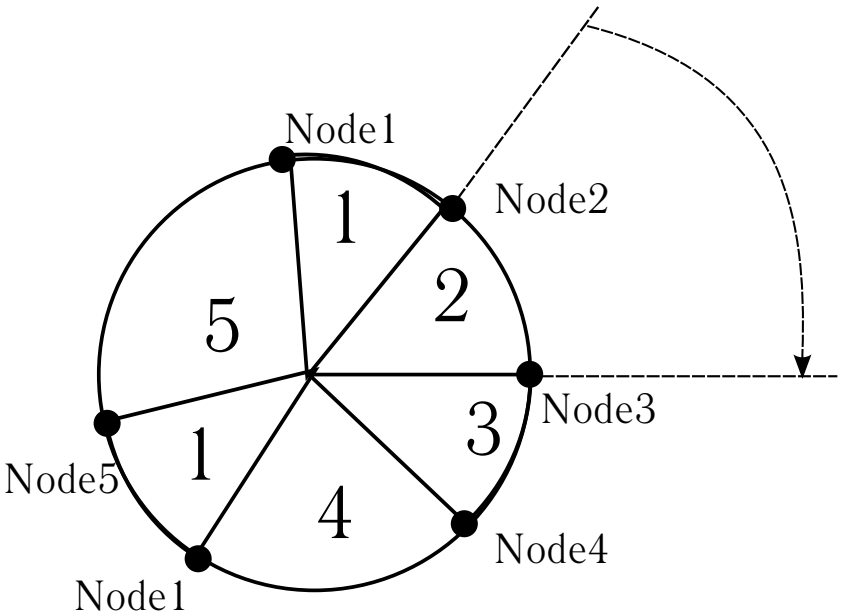
А из нашего окна...

- KeySpace: аналог базы данных
- ColumnFamily: аналог таблицы, имеющей единственный индекс по первичному ключу
- Column: триплет “Имя” – “Значение” – “Timestamp”.

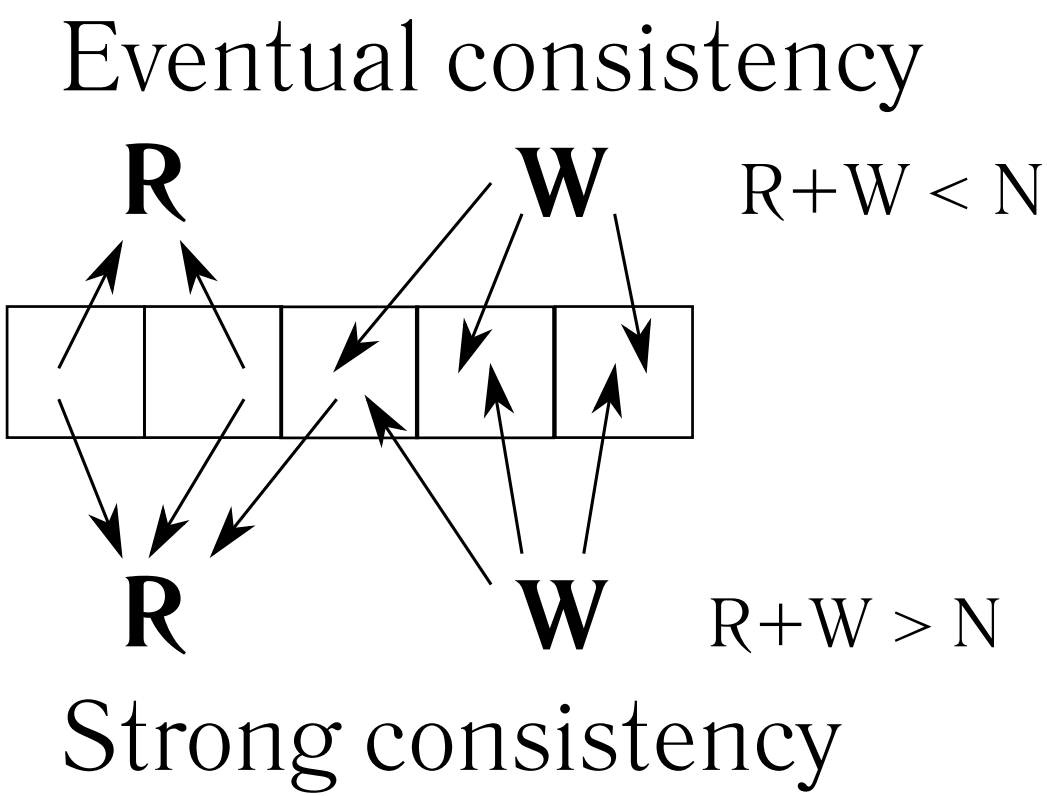


Распределение данных по кластеру

- Использование Consistent Hashing
- Возможность выбора позиции при вводе узла в кластер
- Ручное перераспределение данных



Eventual Consistency



Консистентность будет уже при $R + W > N$, для чего достаточно $R > N/2 + 1$ и $R > N/2 + 1$ — т.н. Quorum reads/writes

SMS Traffic

- Крупный SMS-агрегатор
- Текущая конфигурация: два MySQL-сервера, мастер-мастер репликация
- Нужна возможность как горизонтального, так и вертикального масштабирования

Cassandra у нас

- Используется крупными проектами
- Активно развивается, дружественное сообщество
- Есть примеры реализаций сценариев time-series data, нужных в нашем случае
- Работает на FreeBSD, есть возможность работы из Java и PHP

Проблемы при проектировании схемы

- Отсутствие транзакций
  Но и событий, при которых последовательность действий не удастся, практически не существует :)
- Отсутствие блокировок
  Необходимо использовать внешние решения, например, ZooKeeper, или реализовать сервер блокировок на MySQL.

Проблемы при проектировании схемы

- Отсутствие транзакций
  Но и событий, при которых последовательность действий не удастся, практически не существует :)
- Отсутствие блокировок
  Необходимо использовать внешние решения, например, ZooKeeper, или реализовать сервер блокировок на MySQL.

Проблемы при проектировании схемы

- Отсутствие транзакций
  Но и событий, при которых последовательность действий не удастся, практически не существует :)
- Отсутствие блокировок
  Необходимо использовать внешние решения, например, ZooKeeper, или реализовать сервер блокировок на MySQL.

Проблемы при проектировании схемы

- Отсутствие транзакций
  Но и событий, при которых последовательность действий не удастся, практически не существует :)
- Отсутствие блокировок
  Необходимо использовать внешние решения, например, ZooKeeper, или реализовать сервер блокировок на MySQL.

Вопросы?

- Официальный сайт проекта: http://cassandra.apache.org
- google://
- webmaster@kibab.com, kibab@FreeBSD.org, bakulin@smsmail.ru