Big Data’13

Лекция V: NoSQL СУБД. Google Bigtable

Дмитрий Барашев
bigdata@barashev.net

Computer Science Center

21 марта 2013

Сегодня в программе

Шаблоны доступа к данным

Реляционные СУБД и модные приложения

Таксономия NoSQL

Колоночные СУБД

Bigtable

Анекдот для затравки

A DBA walks into a NOSQL bar, but turns and leaves because he couldn’t find a table

Анекдот для затравки

A DBA walks into a NOSQL bar, but turns and leaves because he couldn’t find a table

then he walks into Google bar and finds a big table there

Сегодня в программе

- Шаблоны доступа к данным
- Реляционные СУБД и модные приложения
- Таксономия NoSQL
- Колоночные СУБД
- Bigtable

Последовательный доступ

- Map-Reduce, Pregel: последовательный доступ и пакетная обработка
- Читается много, записывается много
- Результат используется напрямую или подается следующим звеньям, но не меняется
- Прекрасно для индексирования всего интернета

Другие приложения

- Твиты, шаринги, лайки, котики, ~~ботнеты~~
- Много данных, постоянно добавляются и меняются
- Случайный доступ к данным
- Зато данные достаточно локальны

Варианты хранения

- Классическая реляционная СУБД
  - Oracle, SQL Server, MySQL, etc.
- Модная NoSQL СУБД
  - MongoDB, Cassandra, HBase, etc.

Классическая реляционная СУБД

- Заранее известная структура
- Данные хранятся построчно
- Отношения нормализованы и связаны друг с другом
- ACID транзакции
- SQL, позволяющий выразить очень сложные запросы
- Архитектура, выжимающая на high-end машине всё из сравнительно медленных дисков

Классическая реляционная СУБД

- Заранее известная структура
- Данные хранятся построчно
- Отношения нормализованы и связаны друг с другом
- ACID транзакции
- SQL, позволяющий выразить очень сложные запросы
- Архитектура, выжимающая на high-end машине всё из сравнительно медленных дисков

Это прекрасно и зачастую ненужно

WAT? Это ненужно?

- Если у вас банк или ERP система то ваши данные – это ваши деньги
- Конечно же вам нужны ACID транзакции и нормализация

WAT? Это ненужно?

- Если у вас банк или ERP система то ваши данные – это ваши деньги
- Конечно же вам нужны ACID транзакции и нормализация
- Но никому не интересно, будет на свете котиком больше или лайком меньше
- Зато хочется чтобы сервис был доступен и френд-лента не тупила

Сегодня в программе

Шаблоны доступа к данным

Реляционные СУБД и модные приложения

Таксономия NoSQL

Колоночные СУБД

Bigtable

Строгая схема и нормализация

- *Не дублируй факты! Не допускай аномалий! НФБК! Ключи!*
- Это все так – там где данные стоят денег
- Но абстракции текут уже в РСУБД: клиентские программы объектно-ориентированы, ORM is PITA
- Модные приложения не хотят структуры и нормализации, они хотят ключ-значение

Заранее известная схема

- `ALTER TABLE User ADD COLUMN friend_count INT`

Заранее известная схема

- `ALTER TABLE User ADD COLUMN friend_count INT`
- `ALTER command denied to user '***' for table 'User'`

Заранее известная схема

- `ALTER TABLE User ADD COLUMN friend_count INT`
- `ALTER command denied to user '***' for table 'User'`
- Модные приложения не знают заранее, что им понадобится в будущем
  - или знают, что поле «номер паспорта» в соцсети заполнит не так много пользователей ~~как хотелось бы~~

Сложные запросы

- 30 лет назад перед терминалом сидел оператор и писал запросы на SQL или другом языке
  - он мог написать любой запрос
- Сейчас пользователи смотрят френдленту, жмут лайки и посылают друг другу сообщения
  - запросы модного приложения не отличаются разнообразием
- Значит и движок общего назначения не очень нужен

Одна high-end машина

- К сожалению на IBM System Z у модного приложения нет денег

Одна high-end машина

- К сожалению на IBM System Z у модного приложения нет денег
- Даже если бы они и были, модные приложения не хотят
  - зависеть от одного сервера
  - от одного датацентра
  - от одного континента

Масштабируемость

- Вторую high-end машину так быстро не включишь ~~как минимум нужен банкет~~
- И вообще выгодней арендовать (виртуальные) машины в ДЦ чем держать свои
- Значит СУБД должна работать на low-end железе и считать отказ нормальным явлением
  - а значит реплицироваться и шардироваться

Транзакционная семантика

- Как только СУБД становится распределенной, сразу возникают проблемы с ACID транзакциями и доступностью системы
- Но модному приложению неважно, когда сумма лайков увеличится на 1: в тот момент когда лайк добавлен, или через 5 минут.
- На смену ACID спешит BASE: **B**asic **A**vailability, **S**oft state, **E**ventually consistent

Железо вообще

- 30 лет назад оперативная память стоила дорого; процессоры были слабые
- Сейчас оперативная память измеряется терабайтами и у каждого в кармане по 4 процессорных ядра
- Время позиционирования диска и время доступа к RAM изменилось не сильно ($20ms \rightarrow 4ms$ для диска и $200\mu s \rightarrow 100\mu s$ для RAM)
- Зато появилось три уровня кеша которые существенно быстрее ($\approx 1\mu s$)

Железо вообще

- 30 лет назад оперативная память стоила дорого; процессоры были слабые
- Сейчас оперативная память измеряется терабайтами и у каждого в кармане по 4 процессорных ядра
- Время позиционирования диска и время доступа к RAM изменилось не сильно ($20ms \to 4ms$ для диска и $200\mu s \to 100\mu s$ для RAM)
- Зато появилось три уровня кеша которые существенно быстрее ($\approx 1\mu s$)

Есть место для других алгоритмов

Ситуация не нова: OLAP vs OLTP

On**L**ine **T**ransaction **P**rocessing

- данные обновляются, часто параллельно
- часто затрагивают одну таблицу и почти все атрибуты
- запросы высокоселективны
- ожидается низкое время отклика

On**L**ine **A**nalytical **P**rocessing

- read-only запросы
- у сущностей много атрибутов но запрос интересуется небольшим подмножеством
- низкая селективность
- всякого рода агрегирование
- время выполнения запроса ожидаемо велико

В середине 90-х появились специальные OLAP системы

Сегодня в программе

Шаблоны доступа к данным

Реляционные СУБД и модные приложения

Таксономия NoSQL

Колоночные СУБД

Bigtable

Классификация по модели данных

- «Ключ-значение»: мало чем отличается от хештаблицы, значением обычно является строка
  - тыщи их. Redis из самых популярных
- «Ключ-документ»: значением является документ, который может быть как BLOB’ом так и сложной структурой. СУБД может поддерживать поиск по структуре документа
  - яркий представитель MongoDB
- «Ключ-расширяемая структура»: можно считать разреженной таблицей с бесконечным числом столбцов.
  - Google Bigtable, HBase, Cassandra

Классификация по организации хранения данных

- Построковое хранение
- Колоночное хранение
- Журнальное хранение

Классическое построковое хранение

- все атрибуты кортежа плотно сидят рядом в одном дисковом блоке
- атомарное чтение/запись одного кортежа

Колоночное хранение

- в один блок упаковываются значения одного и того же атрибута разных кортежей
- кортеж разнесен по нескольким блокам
- доступ к одному столбцу существенно эффективнее

Журнальное хранение

- дисковые блоки для чтения read-only
- данные для чтения сидят в RAM
- обновления записываются в RAM и в журнал
- периодически журнал и основные данные объединяются

Сегодня в программе

Шаблоны доступа к данным

Реляционные СУБД и модные приложения

Таксономия NoSQL

Колоночные СУБД

Bigtable

Быстрое получение всех значений атрибута

- Пусть в кортеже в таблице `Webtable` 1000 атрибутов.
- Атрибут `site` – адрес сайта страницы, атрибут `visits` – число посещений сегодня Какая схема хранения выиграет при выполнении запроса?

```
1 SELECT site, SUM(visits) FROM Url GROUP BY site
```

Построчное хранение

site	...+ 500 ...	visits	...+ 500 ...
a.example.com	...	5	...
a.example.com	...	15	...
a.example.com	...	45	...
b.example.com	...	3	...
b.example.com	...	0	...
b.example.com	...	4	...
b.example.com	...	8	...

Колоночное хранение

site	...+ 500 ...	visits	...+ 500 ...
a.example.com	...	5	...
a.example.com	...	15	...
a.example.com	...	45	...
b.example.com	...	3	...
b.example.com	...	0	...
b.example.com	...	4	...
b.example.com	...	8	...

Локальность данных в кеше

- Кеш процессора не такой уж большой
- При построчном хранении будет использоваться малая часть
- При колоночном все нужные значения идут друг за другом – кеш используется эффективнее

Сжатие значений

- Пусть на каждом сайте от 10 до 1000 страниц, в среднем 500.
- Пусть всего миллион сайтов
- Пусть средняя длина адреса 20 байт
- Сколько понадобится места для хранения адресов в построчном хранении и сколько в колоночном?

Сжатие значений

- Строковое хранение

site	...+ 500 ...	visits	...+ 500 ...
a.example.com	...	5	...
и еще 400 страниц			
a.example.com	...	45	...
b.example.com	...	3	...
и еще 400 страниц			
b.example.com	...	8	...

- колоночное хранение

site
a.example.com
+400
a.example.com
b.example.com
+400
b.example.com

Сегодня в программе

Шаблоны доступа к данным

Реляционные СУБД и модные приложения

Таксономия NoSQL

Колоночные СУБД

Bigtable

Bigtable

- Google, первая половина 2000-х годов
- Разреженный распределенный многомерный сортированный хранимый словарь
  - или, ммм... большая таблица со строчками, колонками и версиями значений
- Значение – неинтерпретируемый BLOB
- `(row: string, column: string, time: int64)` $\rightarrow$ `byte[]`

Кортежи таблицы

- Ключ – произвольная последовательность символов
- Кортежи отсортированы в порядке лексикографического возрастания ключей
- Все множество кортежей разбито на *таблеты* – непересекающиеся непрерывные диапазоны
- Таблет – единица распределения работы и нагрузки
- Чтение и запись значений одного кортежа атомарны

Столбцы таблицы

- Семейство: группа однотипных столбцов
- В таблице обычно не очень много семейств но внутри семейства может быть много столбцов
- Группы хранения (locality groups) - набор семейств, которые обычно используются вместе

SSTable

- Сортированное отображение строк в значения
- Сортированный массив с разреженным индексом поверх
- SSTable создается для каждой группы хранения
- SSTable доступна только на чтение
- SSTable живет на GFS

Жизнь системы

- Таблеты обслуживаются таблет-серверами N:1
- В специальной таблице METADATA записано распределение диапазонов ключей по таблетам и таблетов по серверам
- Расположение таблетов METADATA записано в корневом таблете
- Расположение корневого таблета системе известно
- Мастер следит за состоянием таблет-серверов и занимается назначением таблетов

Жизнь клиента

- Клиент читает информацию о том где искать корневой таблет
- Обращается к корневому таблету за информацией о METADATA
- Обращается к METADATA за информацией о том где прописаны таблеты нужной ему строки
- Звонит соответствующему таблет-серверу
- Клиенты кешируют метаинформацию

Жизнь записи в таблете

- Запись производится в redo-журнал, подтверждается и записывается в изменяемый словарь в памяти (memtable)
- Если кортеж удаляется то записывается «могильный камень»
- Minor compaction: когда memtable становится достаточно большим, его замораживают и создают набор новых SSTable

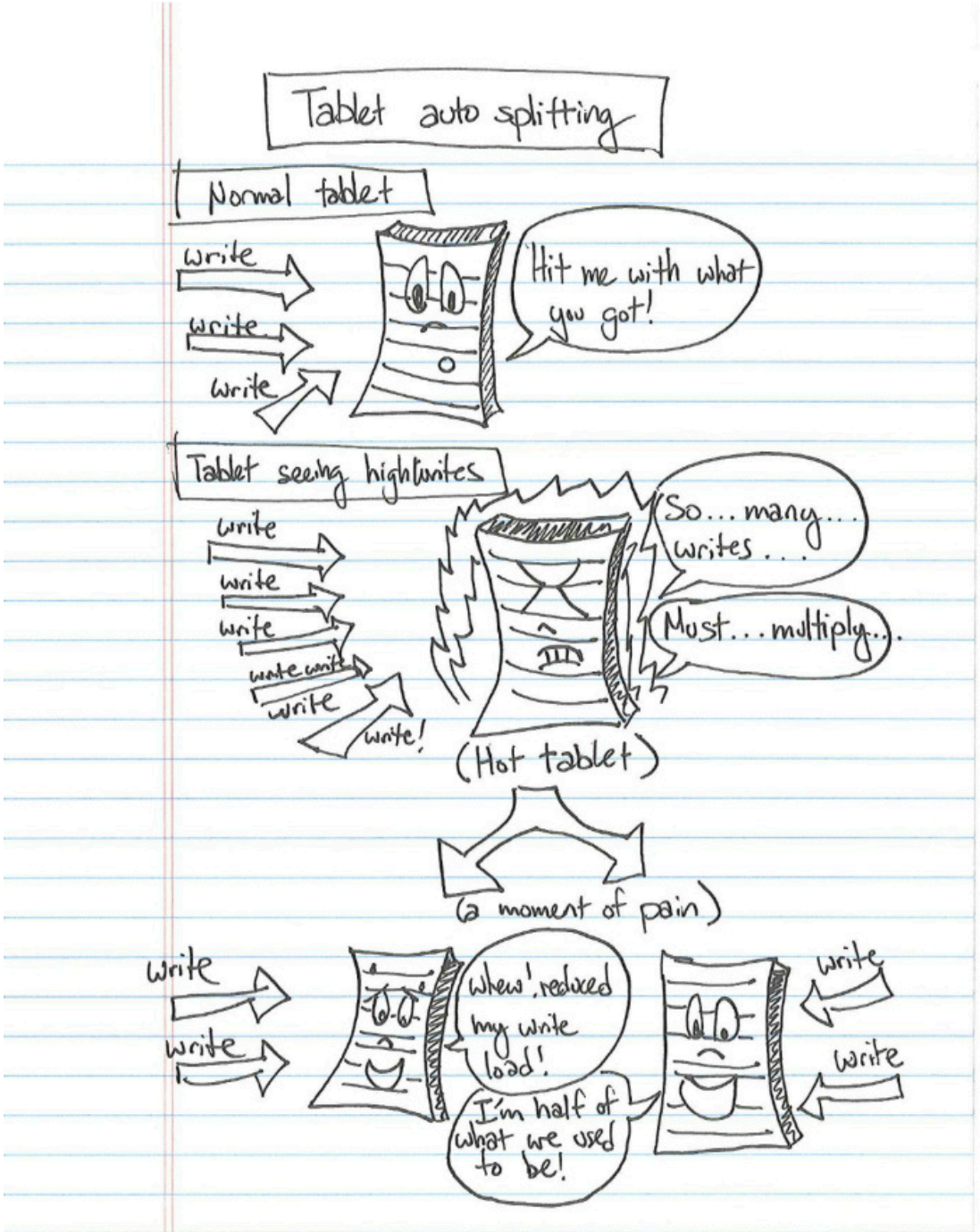
Жизнь чтения в таблете

- Запрос на чтение должен объединить данные из memtable и соответствующих SSTable
- Так как все отсортировано по ключу то это недорогой процесс

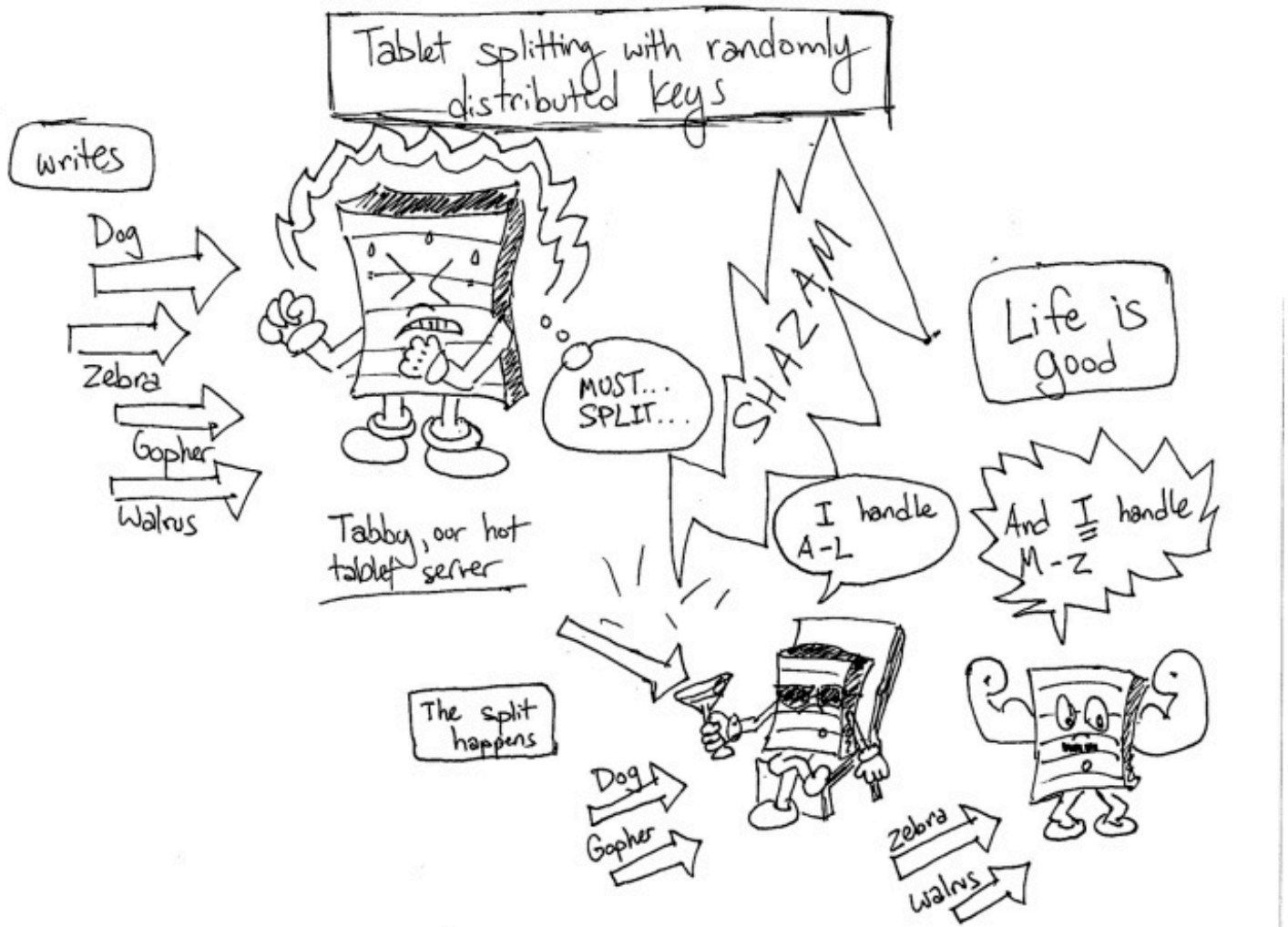
Важные события в жизни таблета

- Разделение (split)
- Большое уплотнение (major compaction)

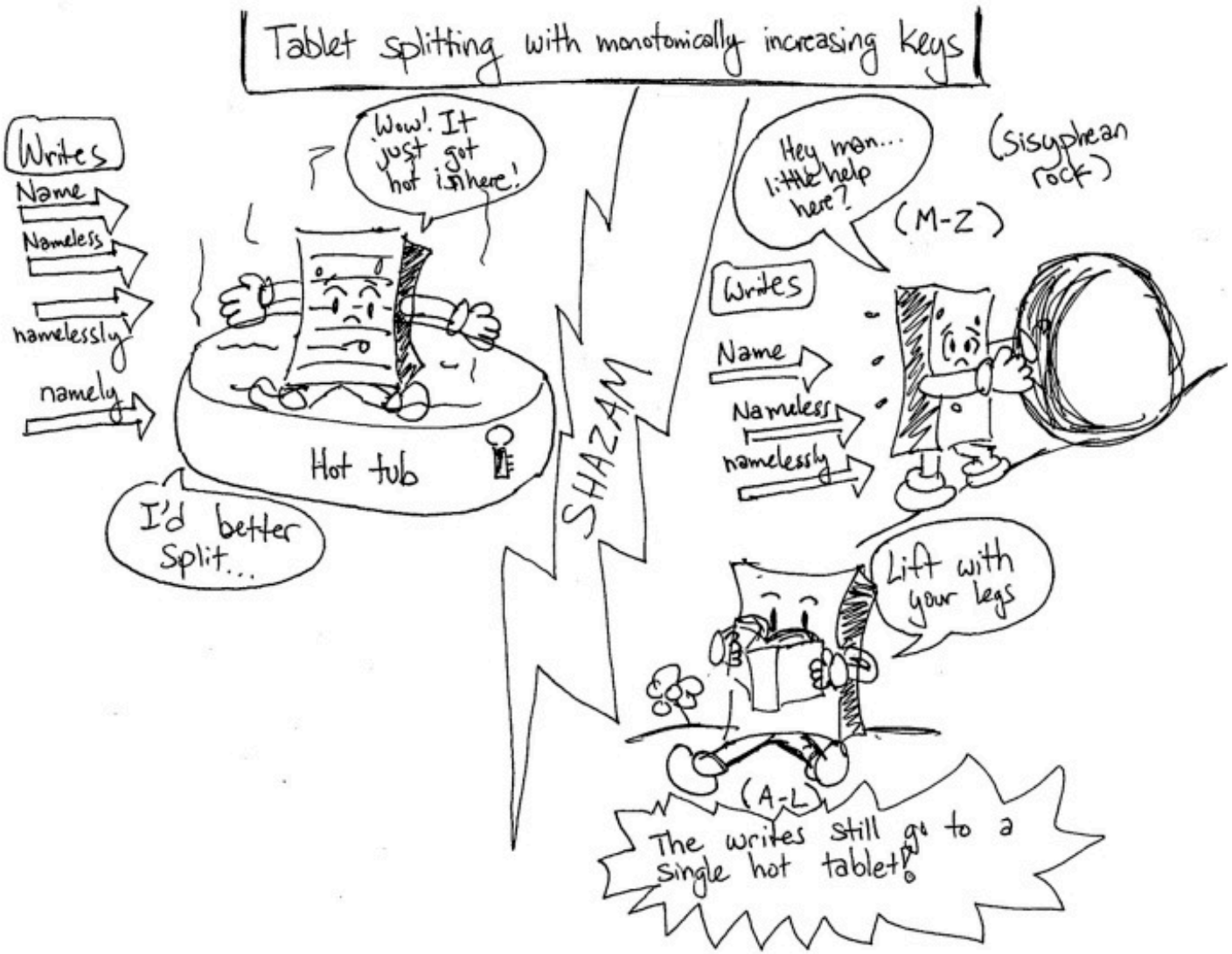
Разделение таблета



Равномерно распределенные ключи



Неравномерно распределенные ключи



Большое уплотнение

- В процессе записи постоянно создаются новые небольшие SSTable
- Время от времени новые и старые SSTable надо объединять
- После уплотнения данные из одного диапазона лежат в одном SSTable и удалены все могильные камни

Реализации Bigtable

- Закрытая в Google (торчит наружу в App Engine datastore)
- Apache HBase Cassandra (родом из Facebook)

Занавес

- Не все большие данные обрабатываются конвейерно
- Не всем приложениям нужны возможности и гарантии реляционных СУБД
- NoSQL СУБД осблуживают приложения с другими требованиями
- Google Bigtable – одна из самых ранних NoSQL СУБД
- Реализаций NoSQL СУБД очень много
- Традиционные СУБД тоже начинают поддерживать NoSQL фичи
- А NoSQL СУБД начинают поддерживать ACID транзакции

Эта презентация сверстана в

LaTeX в вашем браузере
alpha.papeeria.com

Литература I

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber.
Bigtable: A distributed storage system for structured data.
ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008.

Ikai Lan.
App engine datastore tip: monotonically increasing values are bad.
http://ikaisays.com/2011/01/25/app-engine-datastore-tip-monotonically-increasir 2011.
[Online; accessed 21-March-2013].

Литература II

Christof Strauch, Ultra-Large Scale Sites, and Walter Kriha.
Nosql databases.
Lecture Notes, Stuttgart Media University, 2011.