ОНЛАЙН-ОБРАЗОВАНИЕ

Не забыть включить запись!

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Правила вебинара

Активно участвуем

Задаем вопрос в чат или голосом

slack

Off-topic обсуждаем в Slack #канал группы или #general

Вопросы вижу в чате, могу ответить не сразу

OTUS

Занятие 12. Хранилища NoSQL. Назначение и особенности.

Задорожный Антон

Давайте обменяемся опытом

- Выберите реляционную и нереляционную СУБД которые лучше всего знаете
- Оцените свое владение по 10-бальной шкале: 0 – совсем не знаю, 10 – знаю почти как авторы этой СУБД
- Напишите в чатик: СУБД - бал
- Мой пример: Teradata – 9, HBase – 9

Цели вебинара

После занятия вы сможете:

1. рассказать об истории и причинах появления NoSQL СУБД

2. разделять разные виды и некоторых представителей NoSQL СУБД

3. принять решение об использовании такой СУБД в своем аналитическом проекте

Краткое содержание

- Причины и история появления NoSQL СУБД
- Классификация NoSQL СУБД
- Несколько примеров NoSQL СУБД
- NoSQL СУБД в жизни дата инженера

Причины и история появления NoSQL СУБД

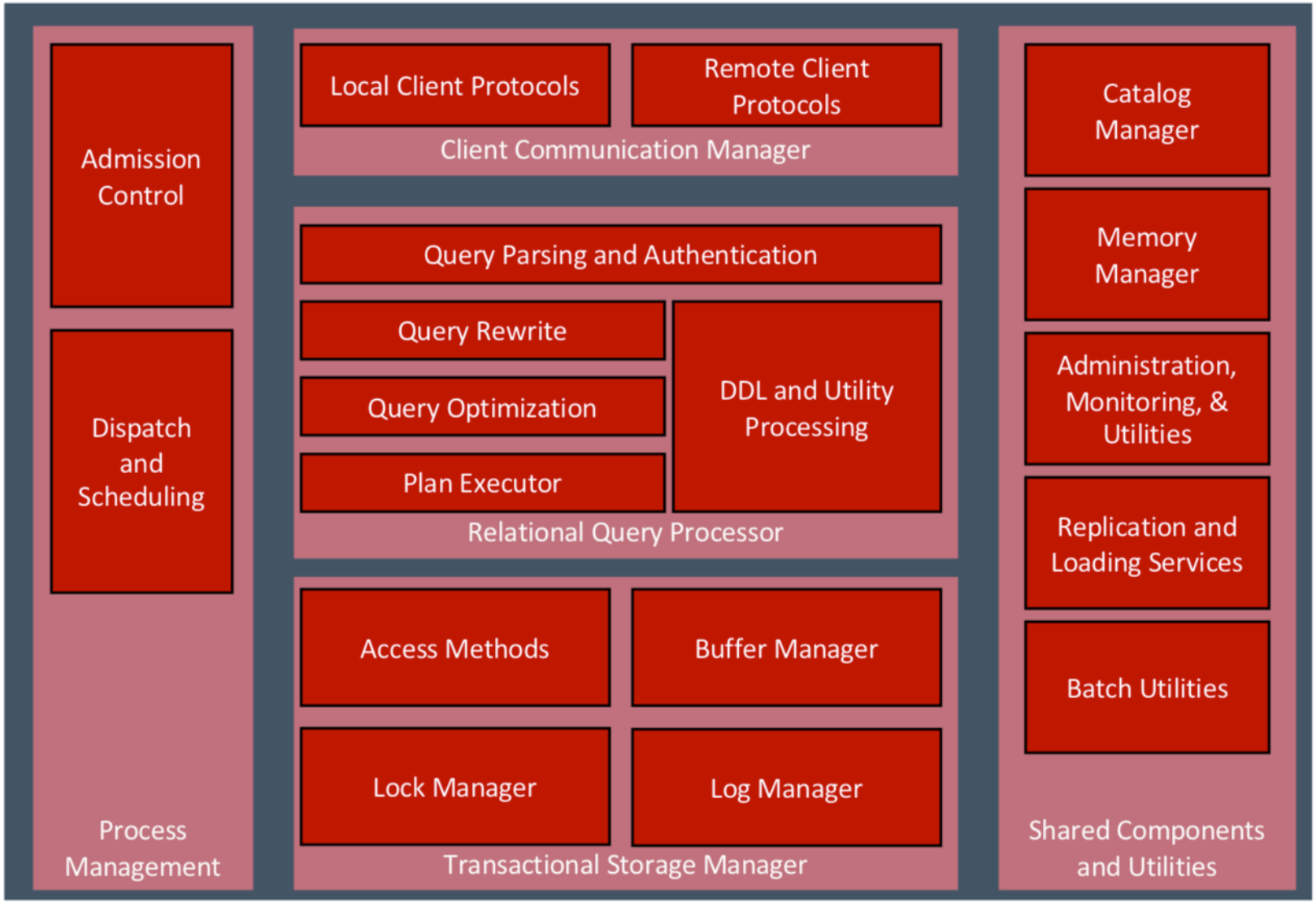
What? No SQL?

OTUS

- Изначально ”non SQL” или “non relational” эти СУБД давали обещания в обмен на отказ от реляционной модели
- Трудно вспомнить другой пример продвижения класса продуктов на основе отсутствия какой-то функции (которая иногда может быть полезной)
- Грузовой автотранспорт – «автомобили без пассажирских сидений»
- Складские помещения – «офисы без мебели и окон»

- NoSQL движение началось как ответ на доминирование реляционных СУБД, проблемы масштабируемости и производительности (а также стоимости) в начале 2000-ых

Наша усредненная реляционная СУБД

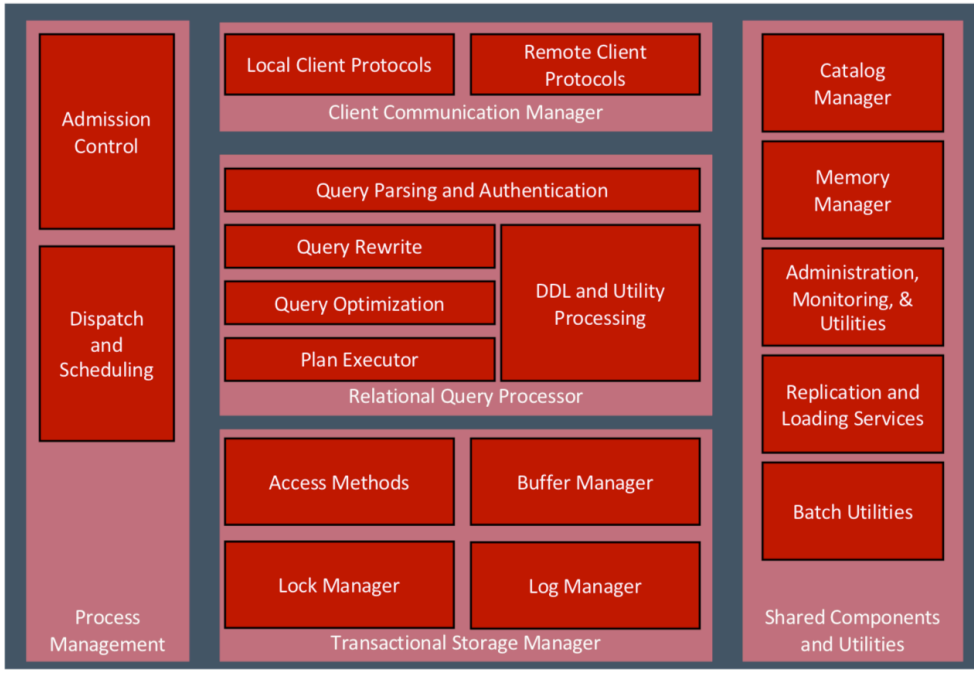


/Joseph M. Hellerstein, Michael Stonebraker, and James Hamilton:
Architecture of a Database System. Foundations and Trends in Databases 1(2): 141-259 (2007)/

Поставим цель – добиться масштабируемости

С точки зрения дизайна NoSQL СУБД это другой набор компромиссов по сравнению с преобладающими в отрасли:

- Фокус на масштабируемости
- Примитивные механизмы обеспечения целостности -> вынос этих задач на сторону приложения
- Примитивная / отсутствующая модель данных -> вынос модели на сторону приложения



Другие причины

- Сложность применения РСУБД для задач хранения графов, документов, кэширования, low-latency нагрузки
- Также в начале 2000-ых ландшафт data management систем был очень беден: существовало несколько нереляционных систем таких как MUMPS или Cache, Berkley DB, но в целом большинство разработчиков использовало либо файлы напрямую (ISAM и схожие концепции) либо реляционные СУБД

- Отсюда появление NoSQL СУБД как отдельного вида, ортогонального РСУБД

Краткая история

- Strozzi NoSQL – 1998
- Google Bigtable – 2005
- Amazon Dynamo (не путать с DynamoDB) – 2007
- MongoDB – 2007

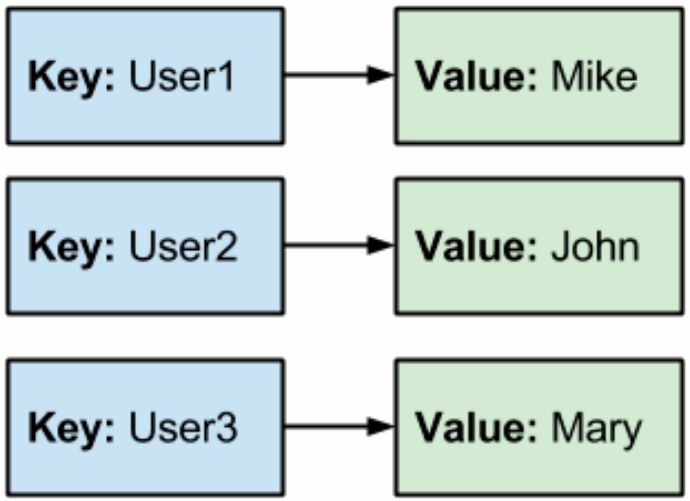
Классификация NoSQL СУБД

Классификация NoSQL СУБД

- Классификация не претендует на полноту
- Почти любая СУБД имеет функции которые могут быть приписаны к нескольким классам
- СУБД развиваются, поэтому моя классификация может отставать – я руководствуюсь собственным опытом и core features этих проектов

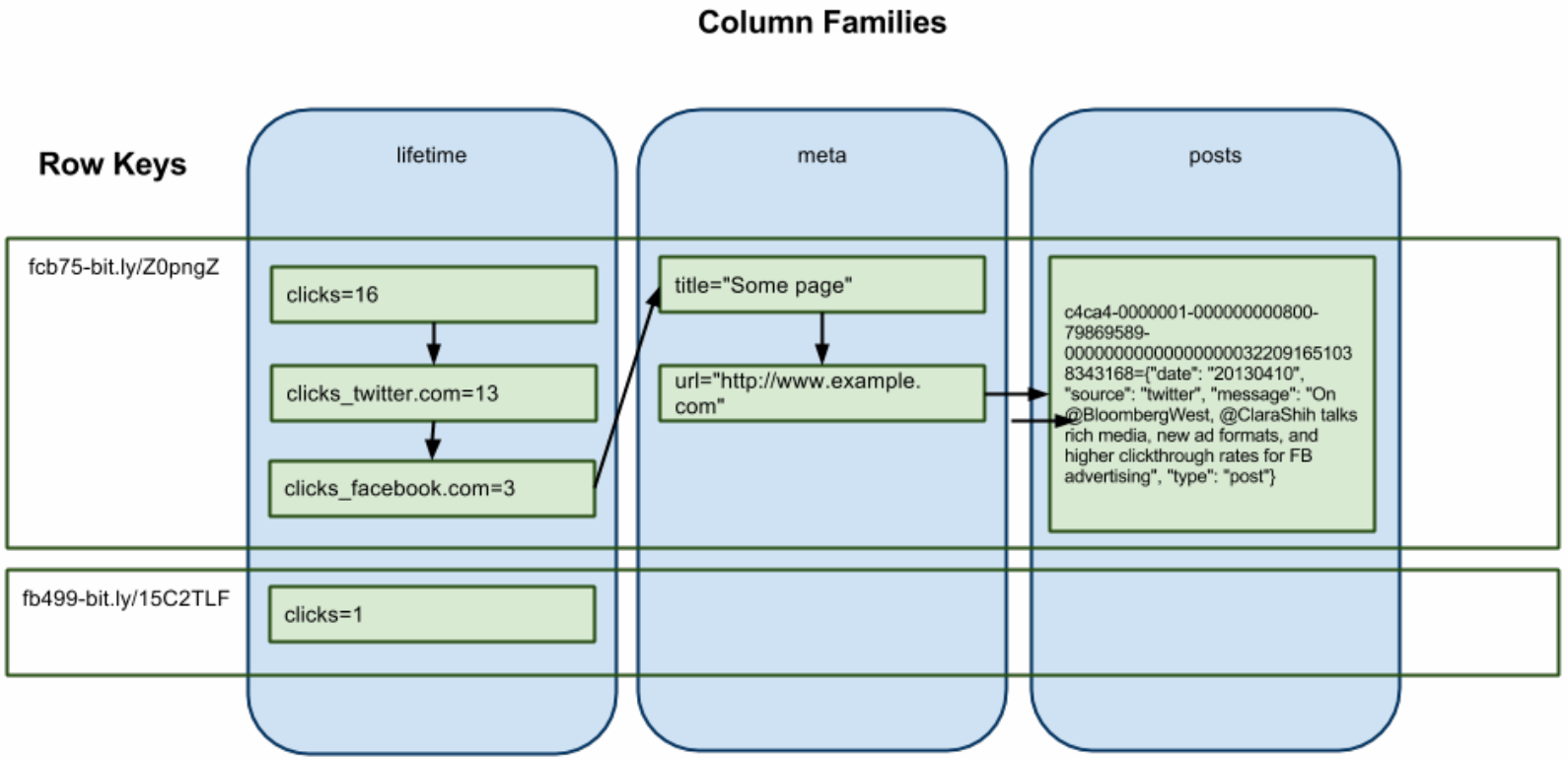
Key-value store

- Распределенные хэш—таблицы для хранения произвольных объектов
- Практически любая СУБД с произвольным доступом по ключу может быть представлена как KV
- In-memory first: Redis, **Aerospike**
- Persistent first: Riak, Dynamo, Oracle NoSQL Database



Wide-column store

- KV внутри KV, хранение произвольного числа колонок внутри значения
- Хорошо подходит для гибкого хранения очень денормализованных объектов
- Google Bigtable, **Apache HBase**, Amazon DynamoDB, **Apache Cassandra**
- Нельзя путать колоночное хранение (columnar storage) с хранилищем с семейством столбцов (wide-column store)

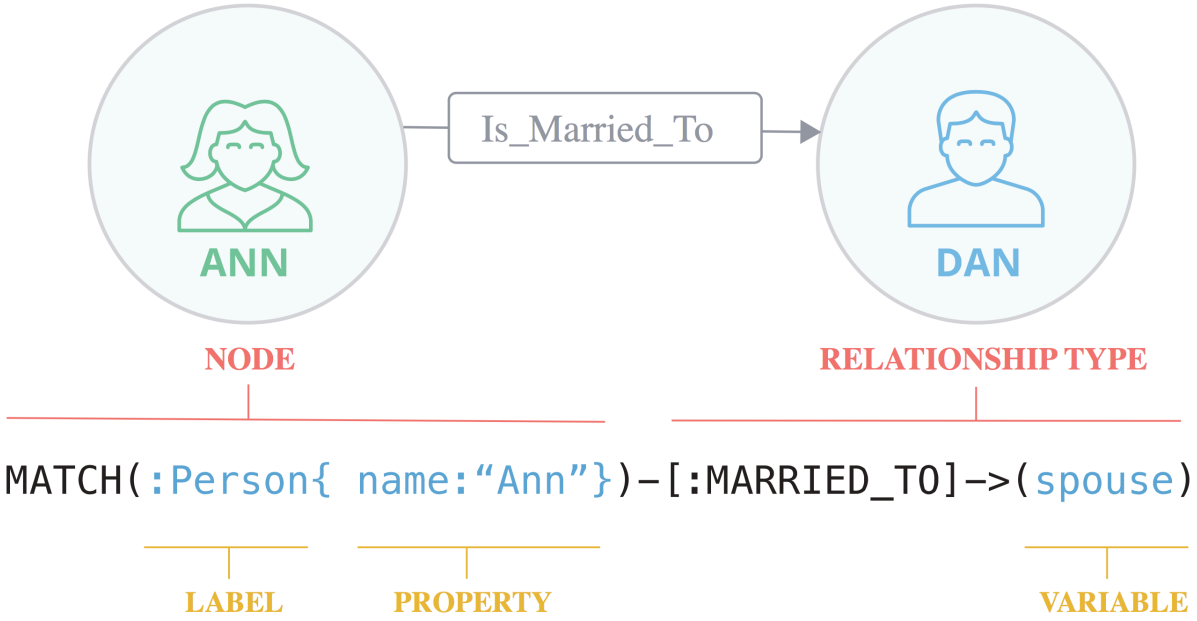


Document store

- Хранилище «документов» – слабо структурированных объектов, например объектов JSON
- MongoDB, Couchbase
- Full-text search: **Elasticsearch**, Apache Solr

Graph database

- В отличии от других NoSQL СУБД графовые базы усложняют модель и требуют особенных подходов
- Часто имеют свои собственные языки для работы с данными, примеры: Cypher (Neo4j), Gremlin (Tinkerpop)
- Идеальная иллюстрация трактовки NoSQL как Not only SQL
- Neo4j, Titan, Apache Tinkerpop*
- Расширения для традиционных РСУБД, т.к. Oracle, Teradata



Итого

- Key-value store
- Wide-column store
- Document store
- Graph database

Несколько примеров NoSQL СУБД

Рассмотрим несколько примеров NoSQL СУБД

OTUS

- Google Bigtable / Apache HBase
- Cassandra / Scylla
- Elasticsearch / Solr
- Aerospike

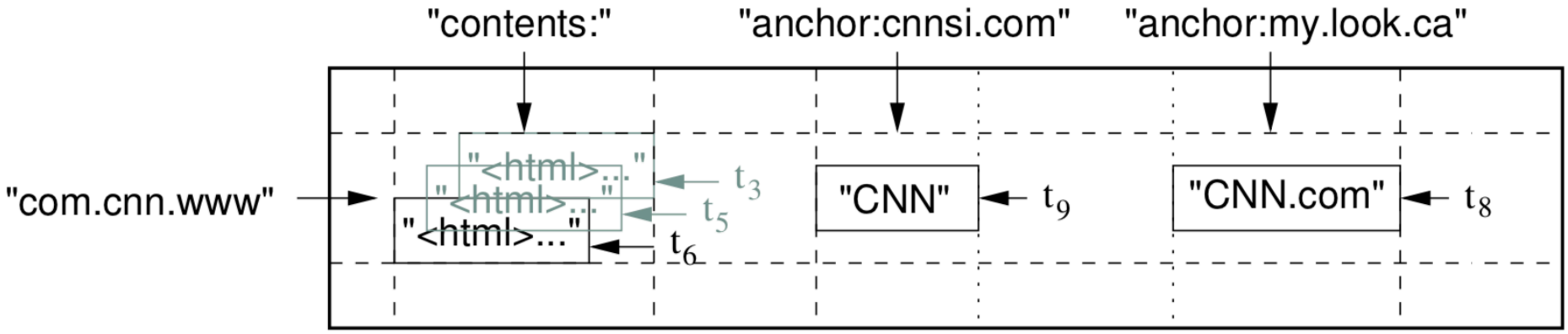
Google Bigtable / Apache HBase

- Google Bigtable, 2004 год, для хранения данных основных сервисов таких как Search, Gmail, Maps
- Активно использует два других внутренних компонента Google: Google File System (GFS, распределенная файловая система) и Chubby (распределенный менеджер блокировок)
- Модель данных: многомерная отсортированная hashmap
- Bigtable: A Distributed Storage System for Structured Data – публикация 2006-го года

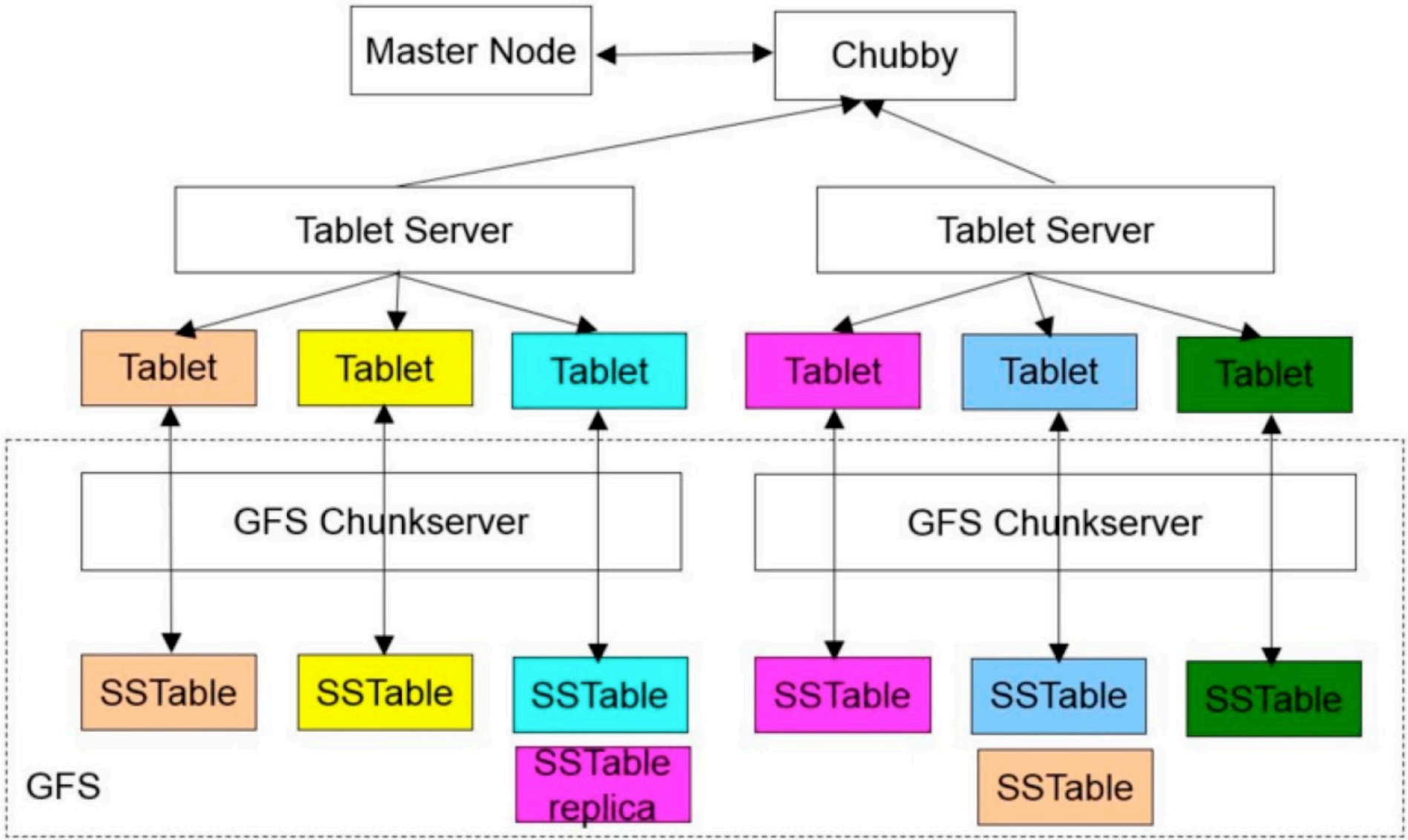
- Apache HBase, 2008 год, реализация по публикации выше (компания Powerset, то что впоследствии стало Microsoft Bing), Java
- Использует HDFS вместо GFS и Zookeeper вместо Chubby
- Совместим с Bigtable вплоть до API (Google Cloud Datastore изначально предоставлял доступ по HBase API)

Модель данных Bigtable

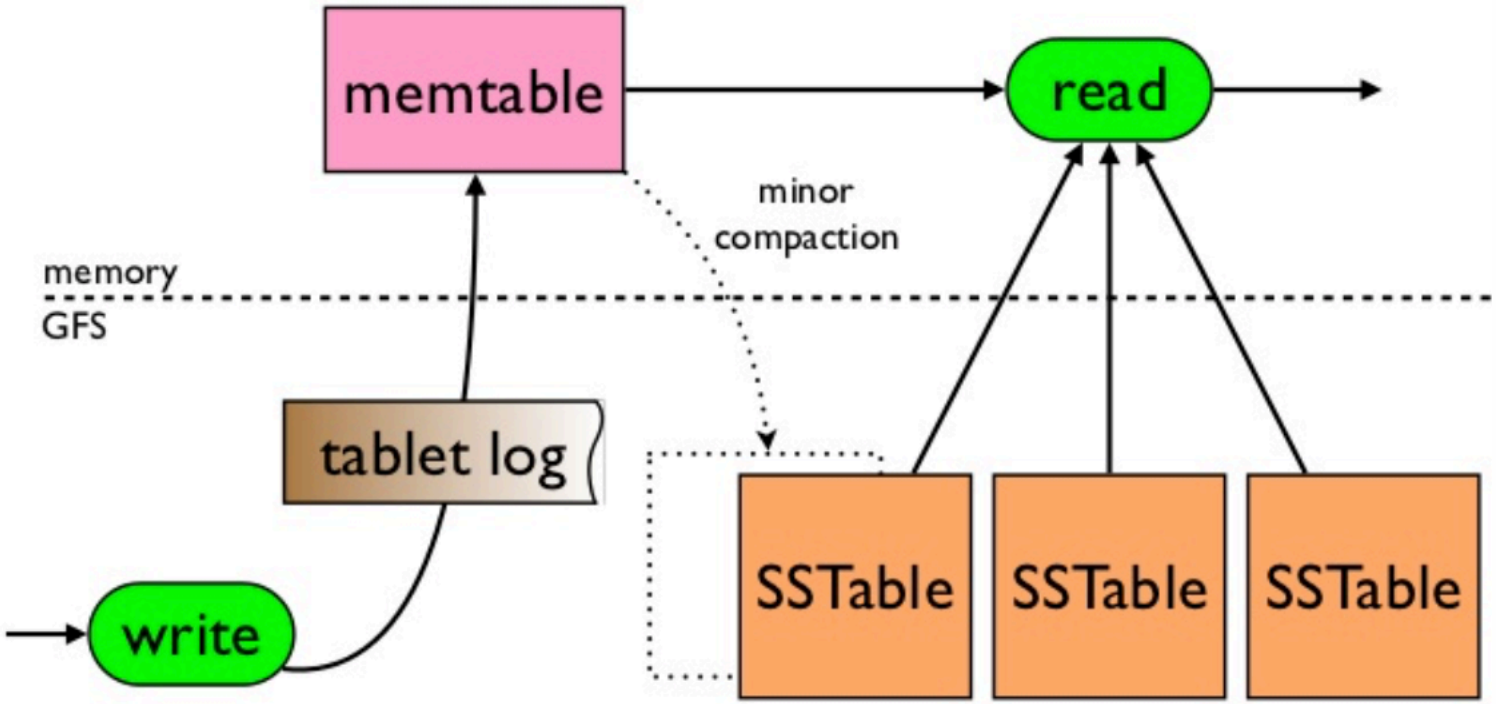
- Rowkey – ключ строки
- Columnfamily – префикс семейства колонок
- Column – полное имя колонки
- t – timestamp времени изменения
- Данные упорядочены по ключу строки!



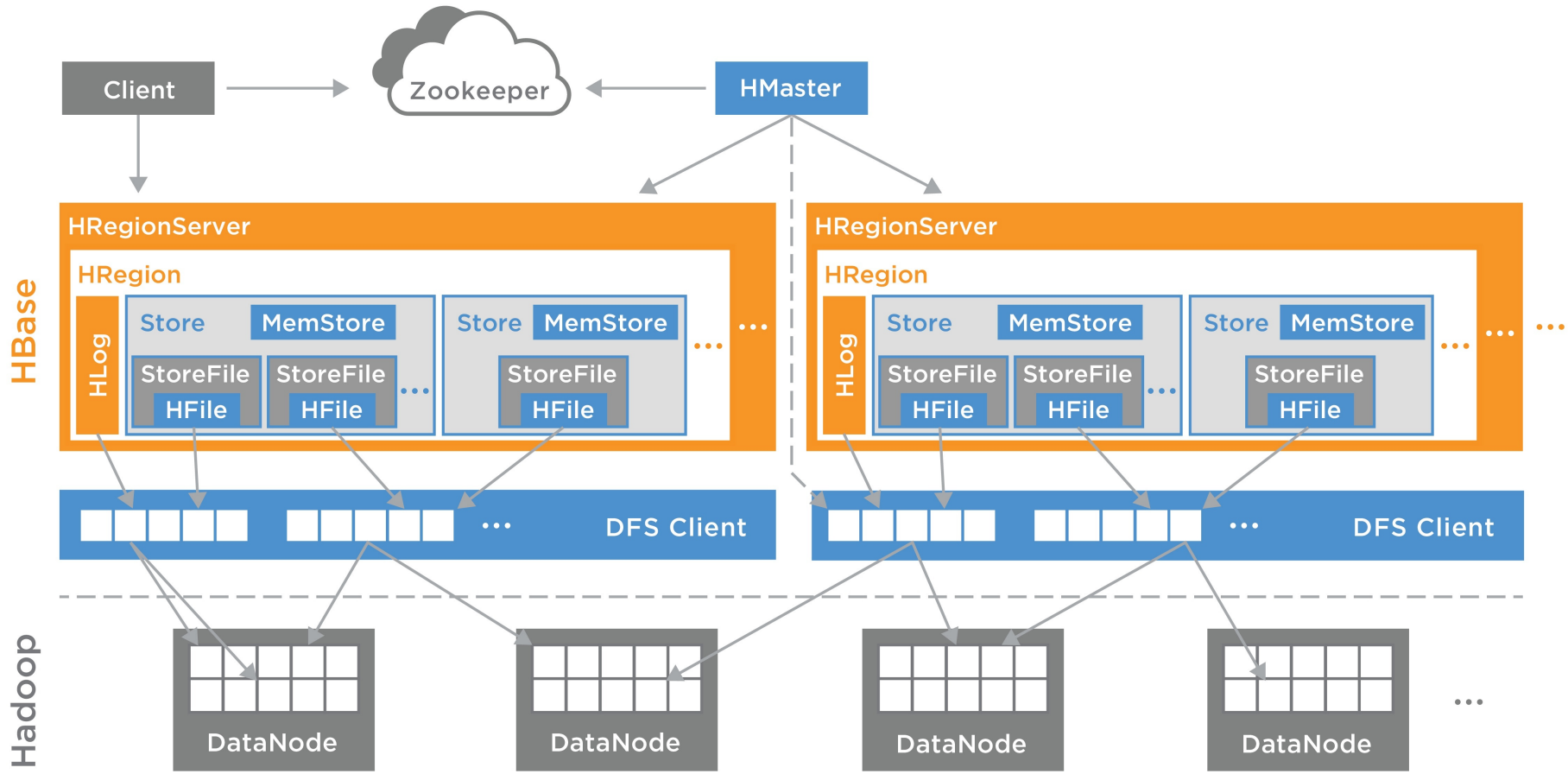
Архитектура Bigtable



Процесс записи в Bigtable



Особенности HBase



Особенности HBase (лучшие практики)

- Если вы делаете это первый раз – скорее всего у вас плохой дизайн rowkey, читайте документацию
- Если вам нужен SQL для OLTP задач – используйте Apache Phoenix
- Если вам нужен SQL для аналитических задач – не используйте HBase (для дашбордов можно использовать Apache Kylin)

- HBase хранит данные в HDFS, HDFS не предоставляет изоляцию IO ресурсов
- Имеет смысл думать о профиле нагрузки на HBase в контексте «типа кластера»: операционный кластер для low latency нагрузки, аналитический кластер для длинных сканов
- Если у вас много чтения и много данных – изучайте BlockCache (on-heap), BucketCache (off-heap), документацию и код, тюнинг кешей творит чудеса

Apache Cassandra

- Facebook Cassandra, 2008 (от авторов Amazon Dynamo), Java
- Wide-column store с некоторыми интерсными особенностями
- Равноправные ноды (без мастера, и без зукипера)
- Встроенная репликация данных (даже со знанием иерархии центров обработки данных)
- Масштабирование и отказоустойчивость с прицелом на always online
- Eventually consistent! (хотя можно выбирать уровень целостности, фундаментально – Last Writer Wins)
- Поддерживается ASF и компанией Datastax

- Scylla – реализация дизайна Cassandra с поддержкой API на языке C++

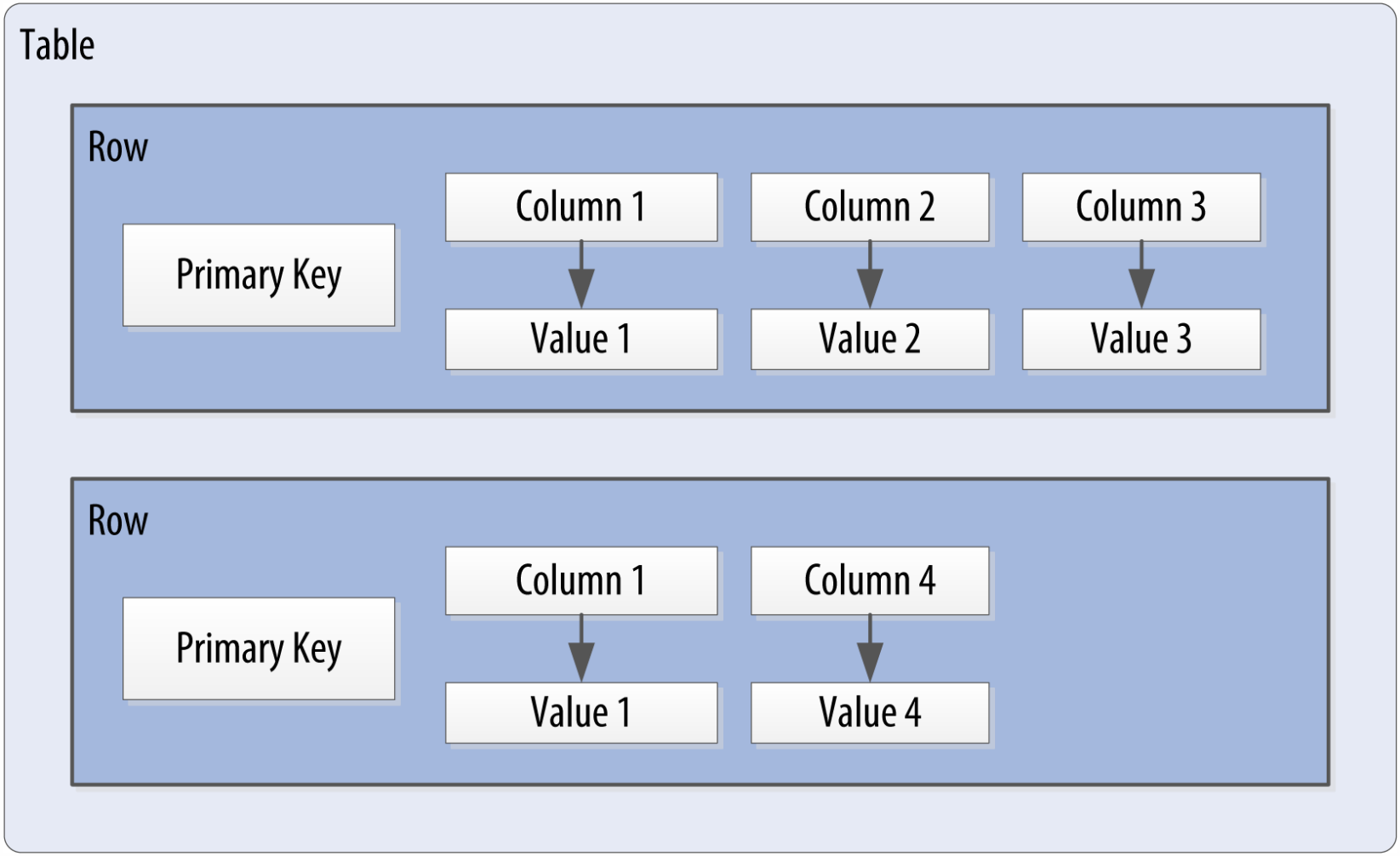
Несколько слов о Dynamo

- Amazon Dynamo – внутреннее eventually consistent KV хранилище в компании Amazon (не путать с DynamoDB)
- Используется в том числе для работы реализации S3
- Вторая важная веха в NoSQL/Bigdata СУБД движении, сформировала такие дизайны как Cassandra, Riak
- В 2007-ом Амазон выпускает публикацию о дизайне Dynamo, которая предлагает решать задачи распределенных систем следующими технологиями:

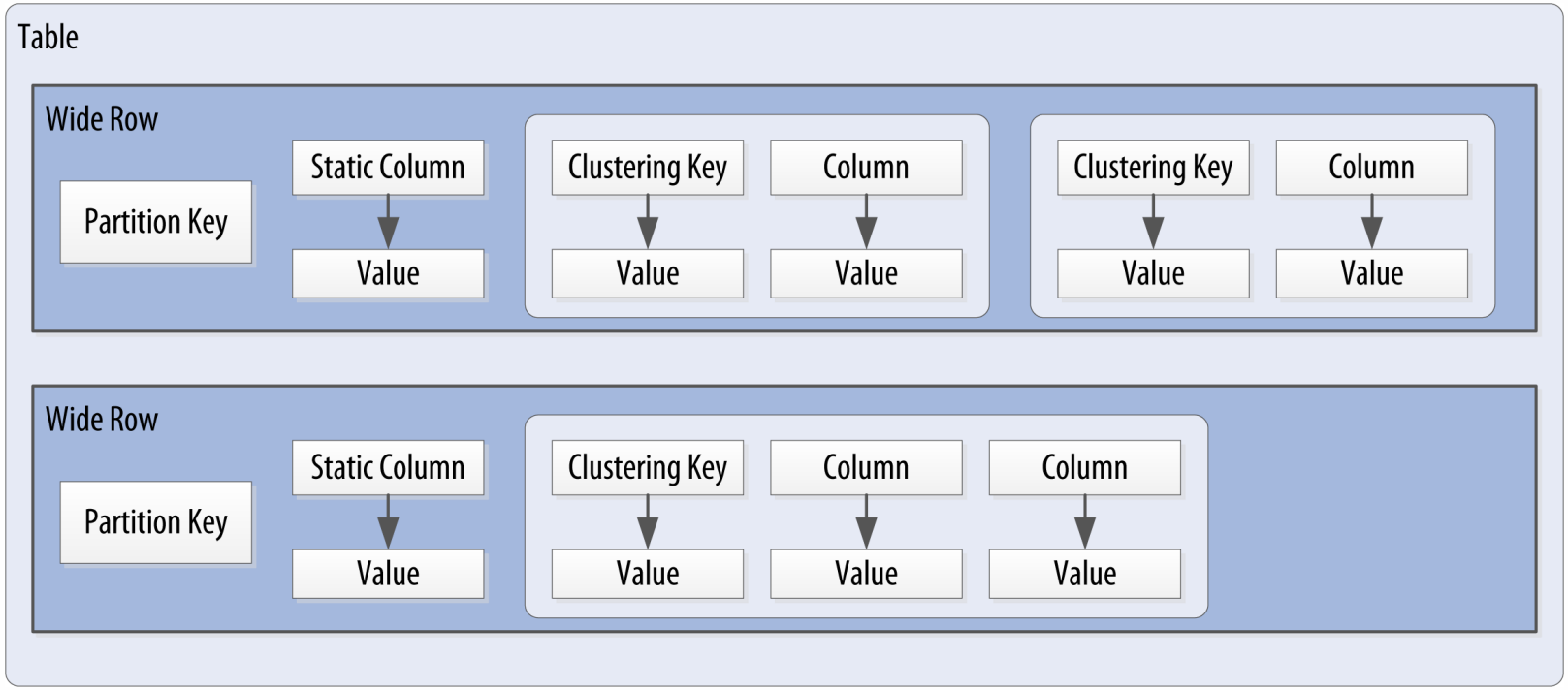
- Разделение данных: предлагается решать при помощи консистентного хэширования
- Высокодоступная запись: обеспечивается алгоритмом векторных часов с реконсиляцей во время чтения *
- Временные сбои: решаются плавающим кворумом (sloppy quorum)
- Восстановление при потере узла: выполняется за счет синхронизации хэш-деревьев Меркла в фоне (anti-entropy)
- Членство в кластере и определение сбоев: gossip-based протокол

*) Не используется в Cassandra

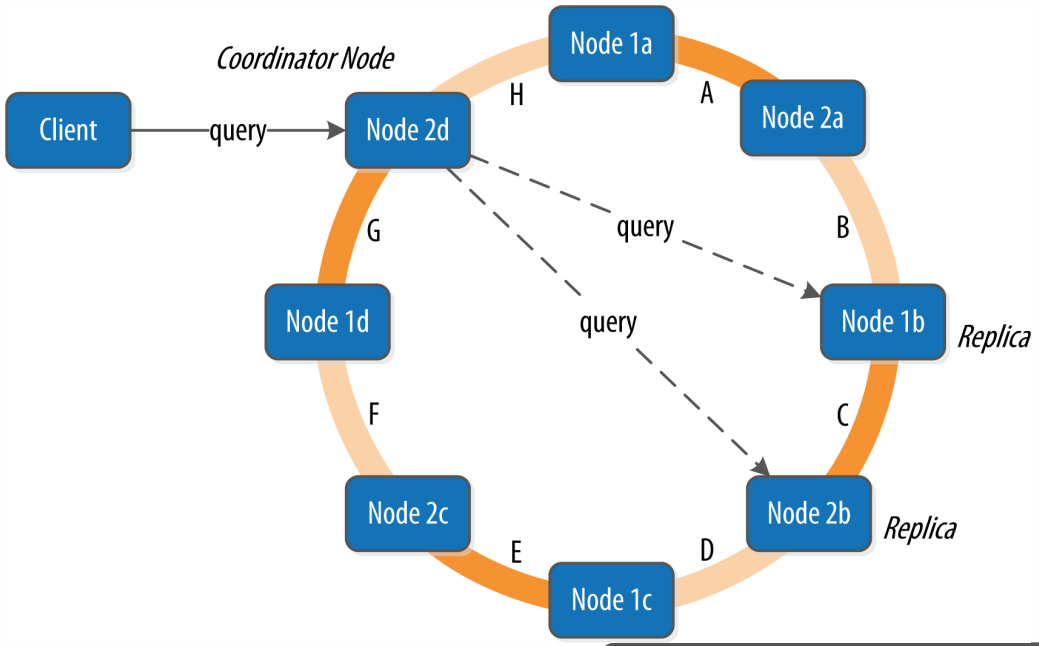
Модель данных Cassandra (простой вариант)

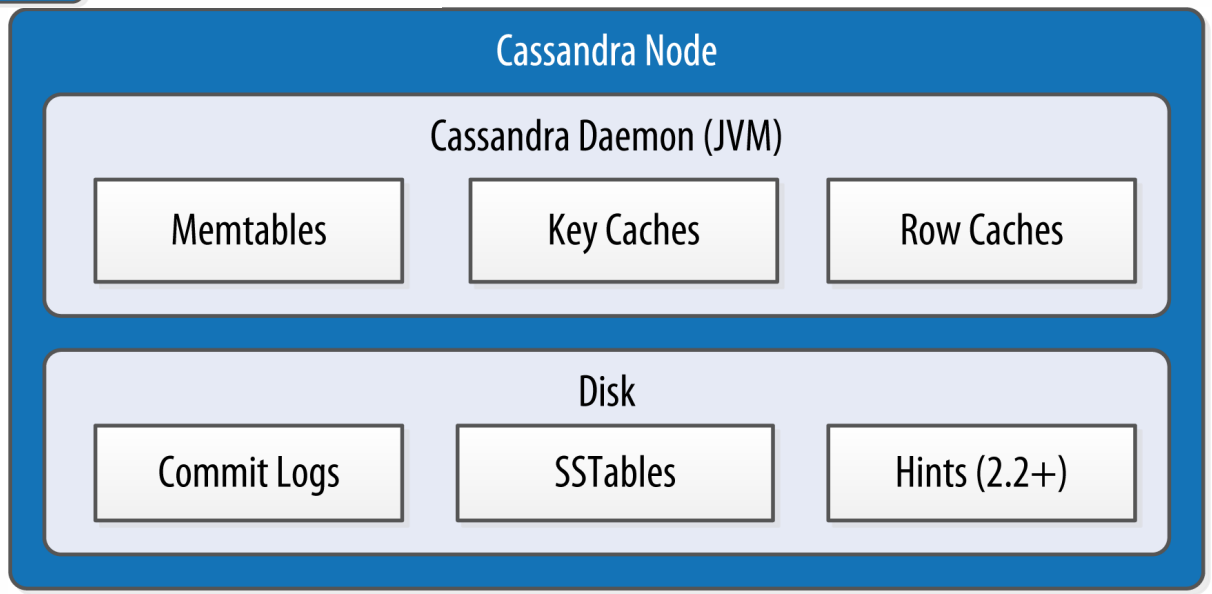


Модель данных Cassandra



Архитектура Cassandra





Особенности Cassandra (лучшие практики)

OTUS

- Из-за богатства возможностей тюнинга (кластеринг, кэши, вторичные индексы) и особенностей реализации модели целостности лучше доверить моделирование данных профессионалам
- Рекомендуемая книга про моделирование и прочие особенности Cassandra: The Definitive Guide, 2nd Edition

- Довольно сложна операционно, опсов не обижать 🤗
- Обратить внимание на то как делается batch load, традиционно это слабое место (использовать bulk loader, а не batch!)

Elasticsearch / Solr

- Elasticsearch, 2010, Java
- Распределенная система полнотекстового поиска на основе библиотеки Lucene
- Распределяет данные при помощи хэша (от задаваемого значения routing)
- Elasticsearch вошел в ELK stack – Elasticsearch Logstash Kibana – очень популярное решение для управления логами и другими слабоструктурированными событиями
- Из-за популярности ELK компания Elastic.co значительно развила непоисковый функционал, в том числе:
- Компонент визуализации Kibana, с разделением пользователей и другими корпоративными функциями
- Функции агрегации результатов
- GET в режиме реального времени (позволяет использовать как NoSQL document store)

- Apache Solr, 2006-2008, Java
- Распределенная система полнотекстового поиска на основе библиотеки Lucene
- Намного меньше заточена под общее хранение, но местами лучше Elasticsearch как индекс

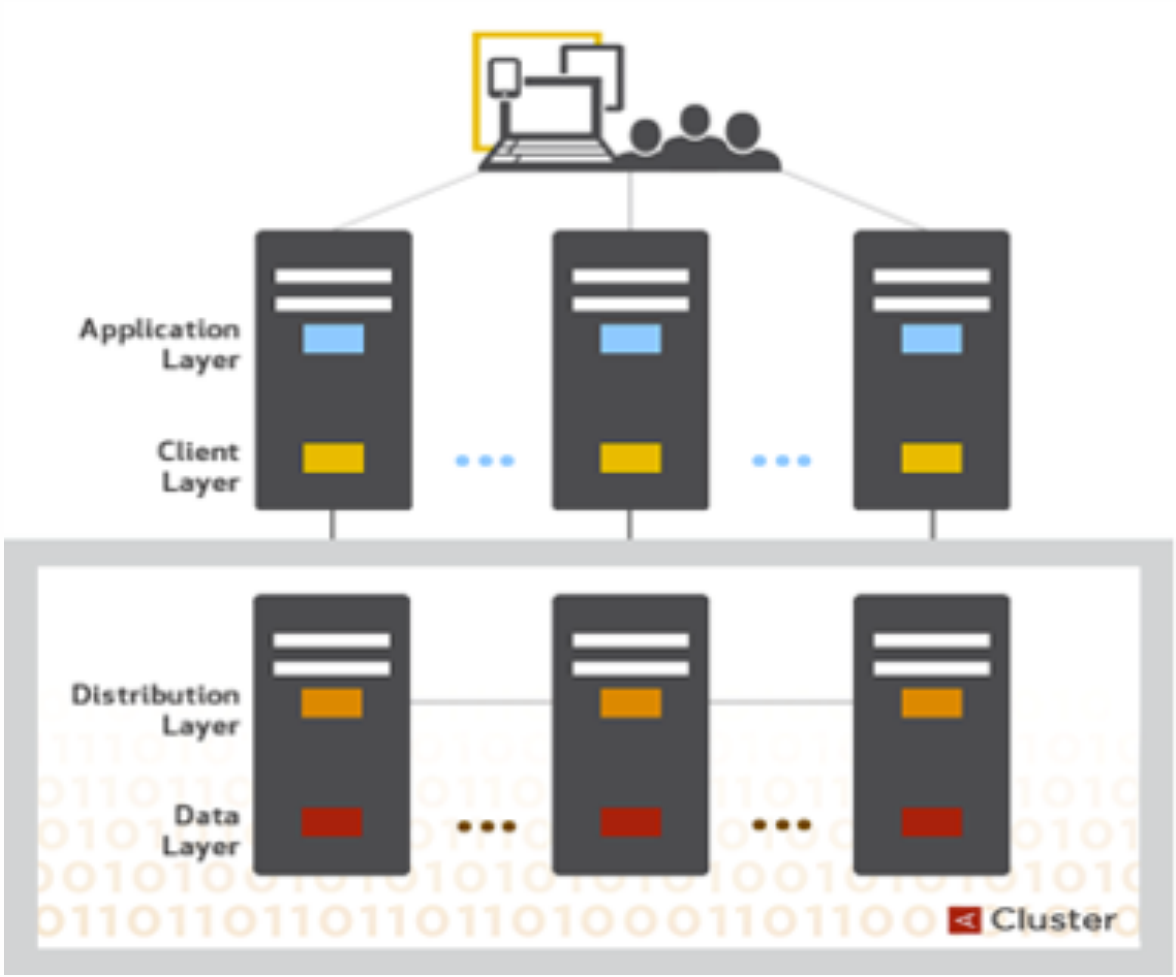
Aerospike

- Aerospike, 2012, ANSI C
- Изначально in-memory key-value store, теперь с персистентностью на флэш
- Развивается активно, но надо смотреть на разные релизы, коммерческий и открытый

Модель данных Aerospike

- Namespace – как БД/схема в обычных СУБД, контролирует где хранятся ключи, время жизни, и количество реплик
- Sets – как таблица в БД, некоторые строки могут не принадлежать ни одному множеству
- Bins – колонки в строке

Архитектура Aerospike



NoSQL СУБД в жизни дата инженера

Для чего нужна NoSQL СУБД дата инженеру?

Представим себе аналитическую систему, она создает data product - некую таблицу, которая используется для получения бизнес-ценность?

1. Отчет / график для человека чтобы принять или подтвердить решение
2. Таблица для другого аналитического процесса (обучение модели, обмен данными)
3. Вектор значений для приложения для каких-то расчетов:
    1. Провести скоринг/предоставить результаты скоринга
    2. Рекомендуемое предложение клиенту
    3. Профиль нагрузки на базовую станцию
    4. История использования конкретного автомобиля

Рассмотри два кейса для NoSQL СУБД:

- Операционализация результатов аналитики / serving layer: выдача вектора значений по конкретному ключу
- Поддержка стриминга

Важные факторы при выборе NoSQL СУБД для serving layer

- Опыт у вашей компании
- Производительность и масштабируемость
- Поддержка пакетной загрузки, наличие интеграций на уровне схем
- Дополнительные требования (вторичные индексы, FTS и прочее)
- Простота

Важные факторы при выборе NoSQL СУБД для поддержки стриминга

По сути требуется OLTP СУБД для конкретного приложения, обычно два вида нагрузки:

- Обогащение события, тогда смотри первый кейс
- Хранение состояния, тогда важны внешние интеграции

В этом сценарии это обычный выбор СУБД под конкретные задачи

Домашнее задание

1. Мы разрабатываем сервис который показывает другим сервисам внутри нашей компании Lifetime Value клиента (по его идентификатору). Мы решили использовать Aerospike в самой простой редакции.

2. Разработайте скрипт на python который содержит такие методы:
   def add_customer(customer_id, phone_number, lifetime_value)
   def get_ltv_by_id(customer_id)
   def get_ltv_by_phone(phone_number)

Срок – 2019.07.29 09:00

Рефлексия

Отметьте 3 пункта,
которые вам
запомнились с вебинара

- Что вы будете применять в работе из сегодняшнего вебинара?

Следующий вебинар

Тема: SQL-доступ к данным. Apache Hive.

Среда 2019.07.17 в 20.00

Ссылка на вебинар будет в ЛК за 15 минут

Заполните, пожалуйста, опрос о занятии

Спасибо
за внимание!

ОНЛАЙН-ОБРАЗОВАНИЕ