Архитектура распределенных систем

Первоисточник: Архитектура распределенных систем - YouTube

План лекции:

1. CAP-теорема
2. Eventual consistency
3. Распределенные СУБД
4. Репликация
5. Восемь заблуждений распределенных систем
6. Идемпотентность
7. Патерны в проектировании распределенных систем

Введение

- Планирование нагрузки
- Требование к надежности (SLA)

SLA (Service Level Agreement) - 90%, 99%, 99,9% и т.д. - соглашение об уровне сервиса, % доступности сервиса в день, неделю, месяц, год.

Service Level Agreement - SLA

Описывает качество услуг в заданный период (день, месяц или год)

SLA (%)	Простой в день	Простой в неделю	Простой в месяц	Простой в год
90 %	2h 24m	16h 48m	3d 1h	5w 1d 12h
99 %	14m 24s	1h 40m 48s	7h 18m	3d 15h 36m
99,9 %	1m 26s 400ms	10m 4s 800ms	43m 48s	8h 45m 36s
99,99 %	8s 640ms	1m 0s 480ms	4m 22s 800ms	52m 33s 600ms
99,999 %	864ms	6s 48ms	26s 280ms	5m 15s 360ms
99,9999 %	86ms	605ms	2s 628ms	31s 536ms
99,99999 %	9ms	60ms	263ms	3s 154ms

1. CAP-теорема

CAP-теорема - в любой системе можно обеспечить неболее двух свойств:

- C (Consistency) - консистентность
- A (Availability) - доступность
- P (partition tolerance) - устойчивость к разделению

Главный вывод CAP-теоремы: **При разделении система или консистентна (C) или доступна (A).**

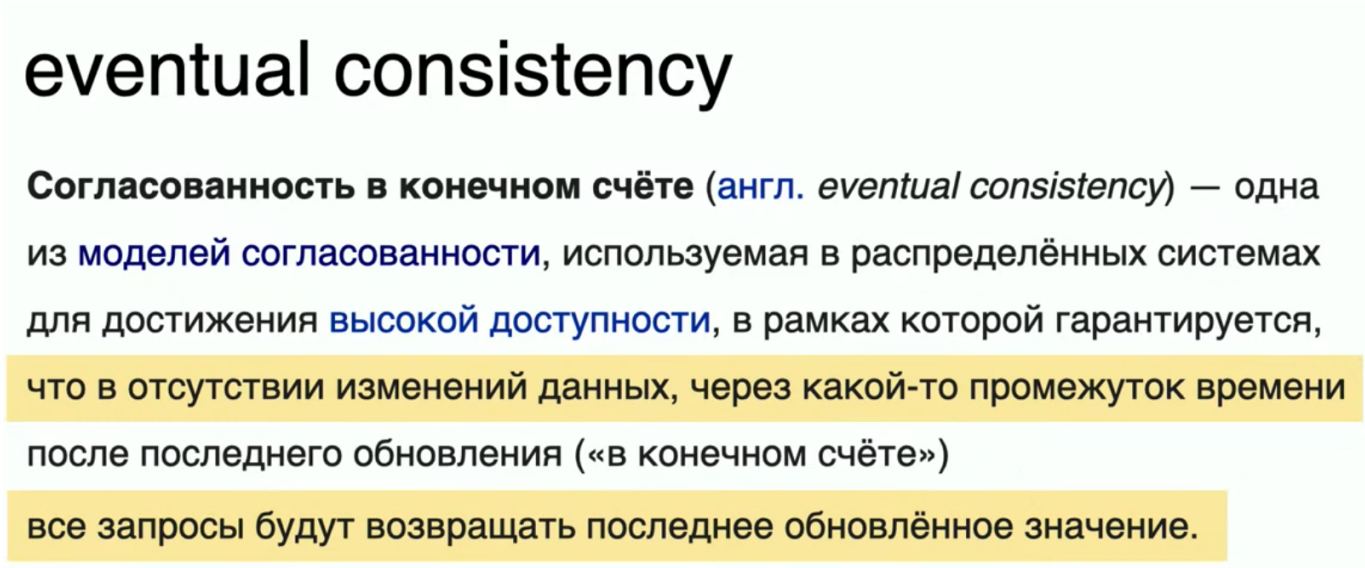
Виды систем:

- CP - отказ (например, PostgeSQL);
- AP - всегда доступна (например, MongoDB);
- CA - не существует.

CA-система только на локальном ноутбуке, как только работает по сети - CA-системы не существует.

2. Eventual consistency

Eventual consistency - согласованность в конечном счете (разрешение конфликтов).



3. Распределенные СУБД

Портицирование - когда данных очень много делаем три PostgreSQL, куда можно записывать данные. В общем случае деление данных на части, эти части можно положить на разные сервера, а можно в разные таблицы. Например, портиция по дням (для заказов).

- Умный клиент (ставить четвертый PostgreSQL - больно);
- Умный сервер (MongoDB - неравномерная нагрузка на серверы).

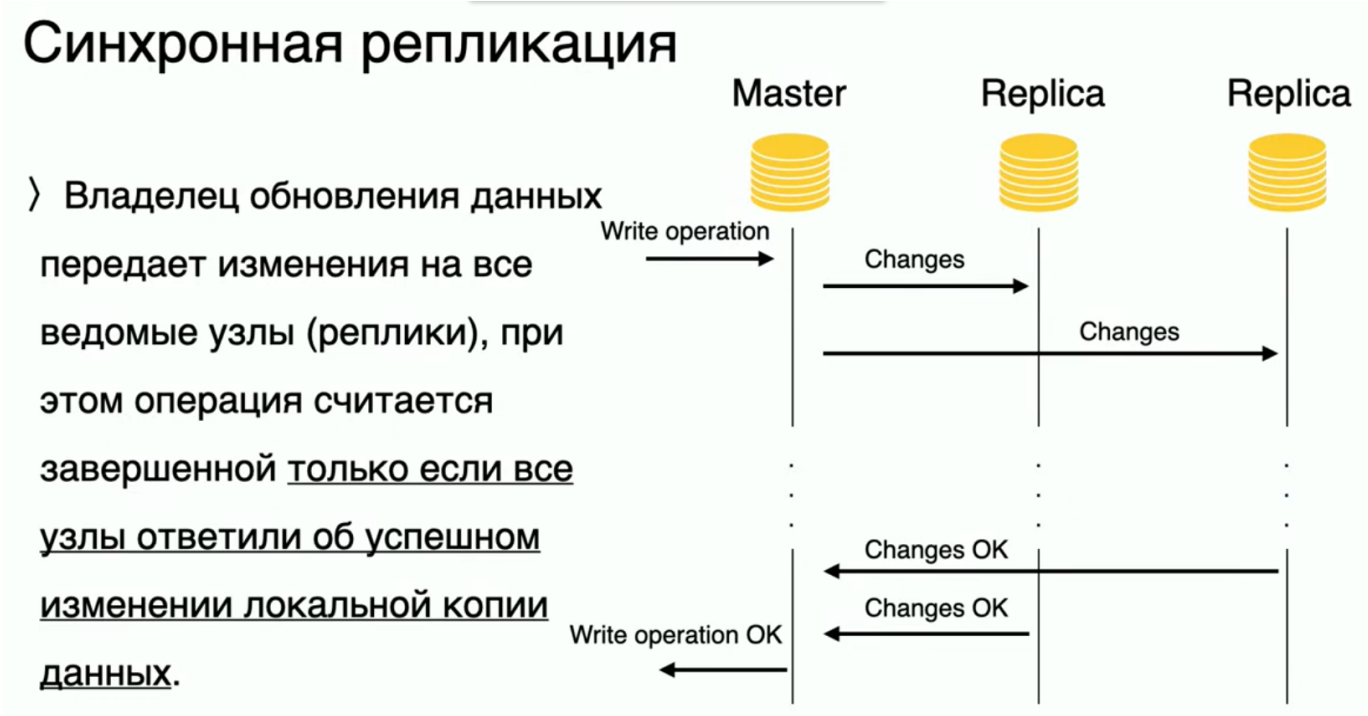
Умный клиент vs. Умный сервер (в зависимости от требований заказчика).

Шардирование - это когда мы раскладываем на разные сервера.

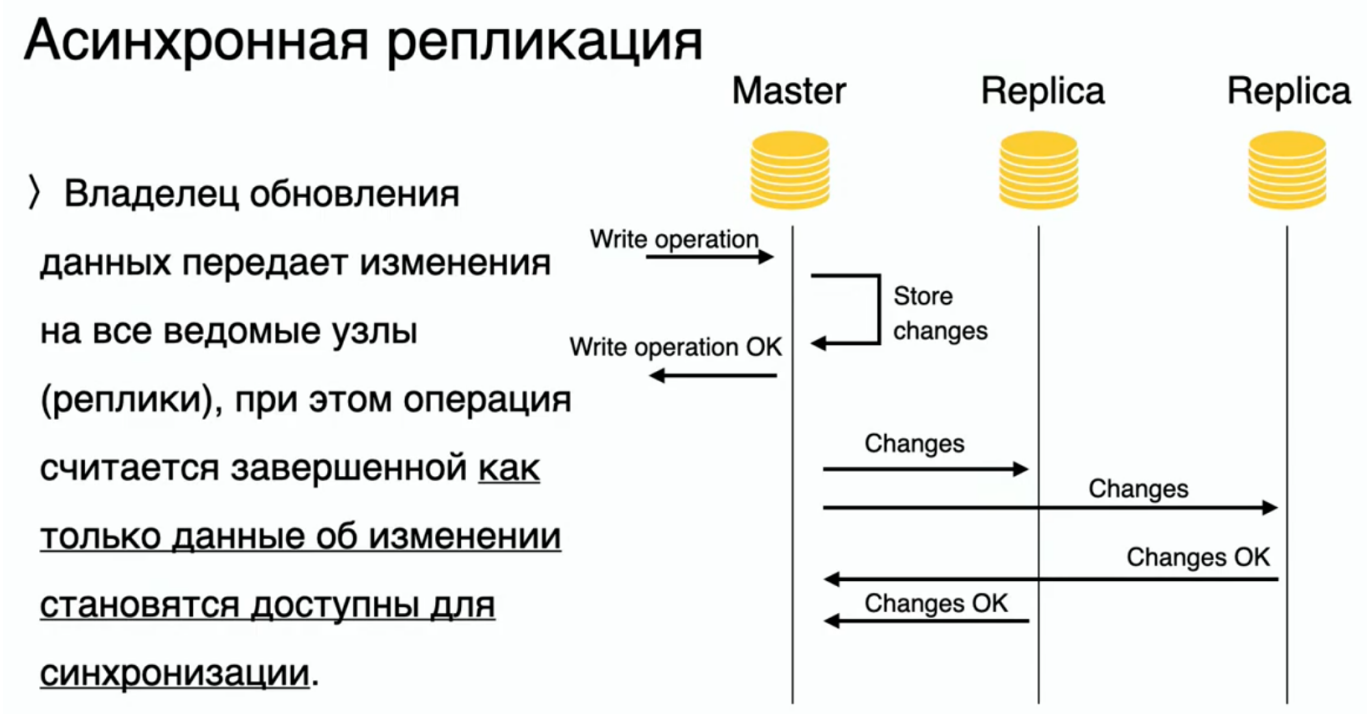
4. Репликация

Репликация - когда данные из одного сервера переходят в другой сервер.

Синхронная репликация - master отвечает только после подтверждения реплик.



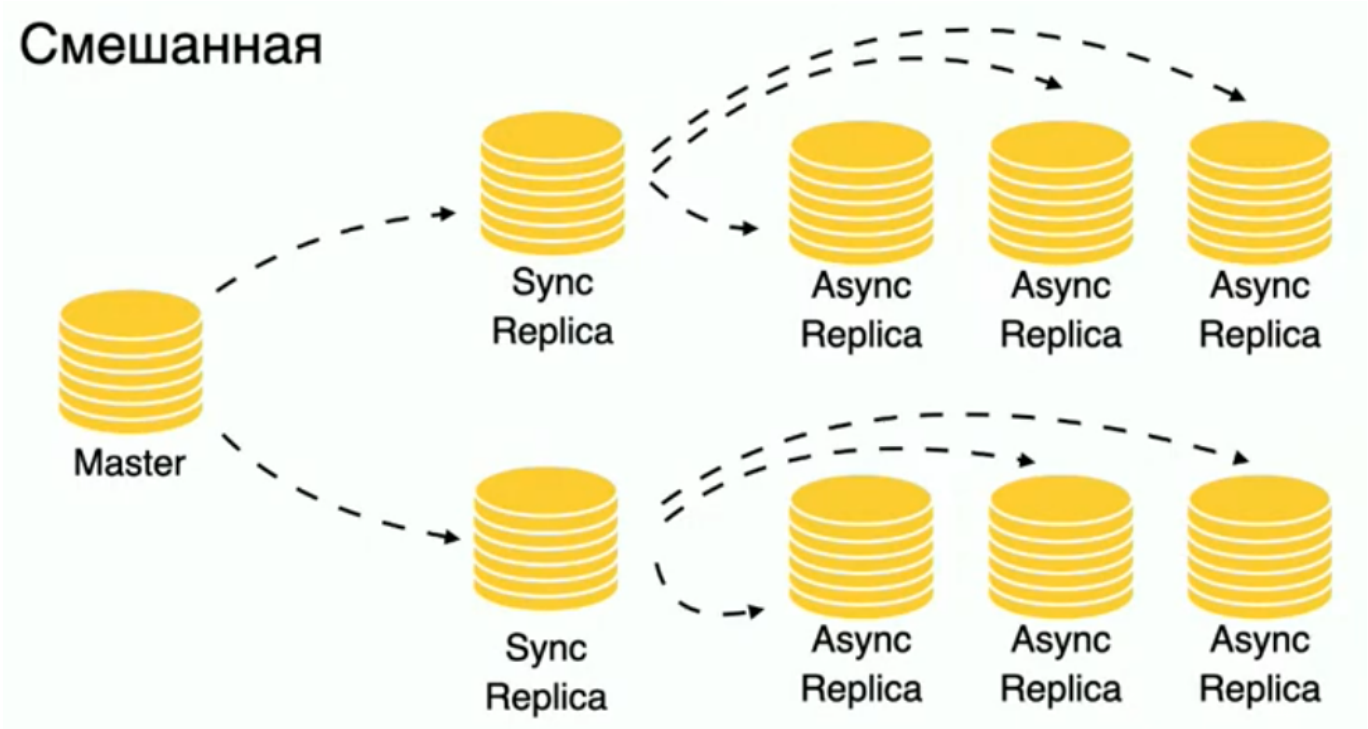
Асинхронная репликация - master отвечает сразу, запись в реплики осуществляется потом.



Смешанная репликация - несколько синхронных реплик и много асинхронных.

Master - почитать/записать.

Реплика - почитать.



Кворум БД - если узлы знают свою роль (master, реплика).

5. Восемь заблуждений распределенных систем

Заблуждения на то и заблуждения, чтобы в них верили (с).

...и что делать?

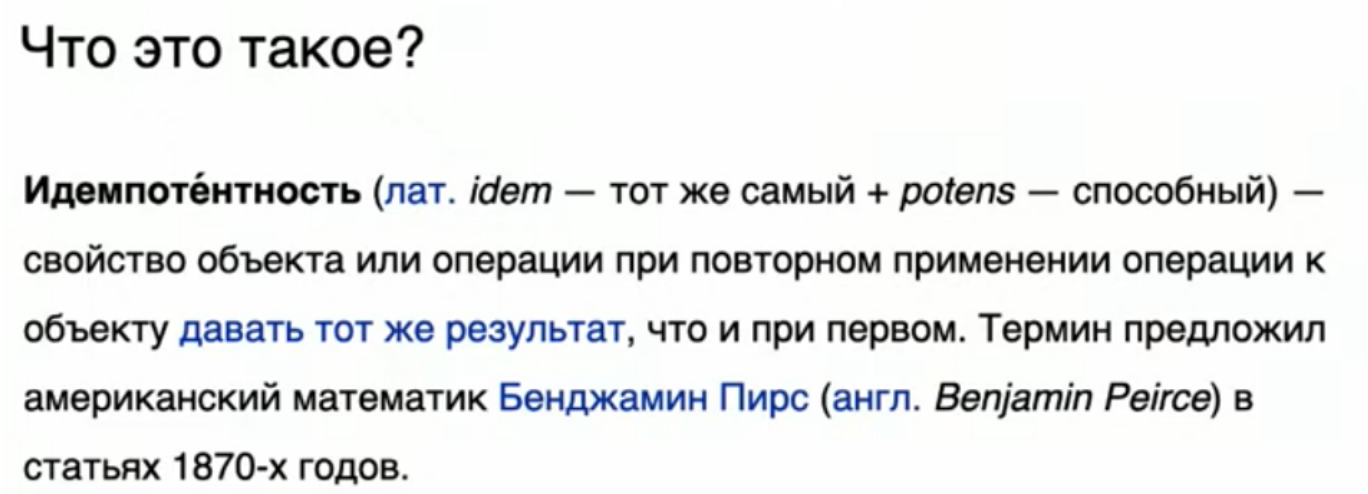
1. Сеть надежна
    - обрабатывать сетевые ошибки
2. Задержка равна нулю
    - не привязывать бизнес-логику ко времени выполнения операции
    - переместить данные ближе к клиенту
    - инвертируйте поток данных
    - верните все данные, которые могут понадобиться клиенту
3. Канал передачи данных очень широкий
    - передавайте только те данные, которые нужны
    - используйте сжатие (http)
4. Сеть безопасна

- шифруйте данные (TLS/SSL - за датацентром)

5. Топология сети не поменяется
    - проектируйте систему так, чтобы повторы запросов не поменяли логику
6. Есть один администратор
    - DevOps - методология разработки ПО
    - infrastructure as code
7. Транспортные расходы равны нулю ($)
    - экономьте трафик
    - мониторинг трафика, соотношение входящий/исходящий
8. Сеть гомогенная (одинаковые уловия у всех)
    - попробовать выполнить эмуляцию в Edge с потерей 1% пакетов

6. Идемпотентность

2+2 = 4, 2*2 = 4 -> сколько раз нне считай, получаем одно и тоже.



Idempotency Key in SOA (Service Oriented Architecture)

7. Паттерны в проектировании распределенных систем

- КЭШ
- Повторы (повторяем запрос пока не получим нужный результат на стороне сервера)
- Предохранитель
- Асинхронные запросы