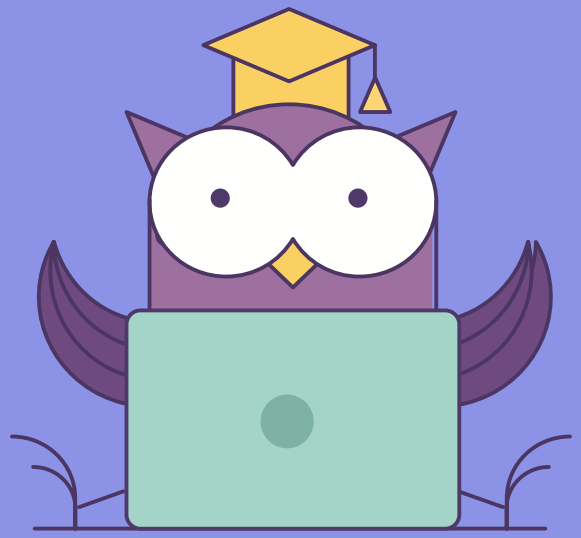
OTUS

CP системы. RAFT, PAXOS, ZAB

Архитектор ПО

Ярослав Астафьев

Меня хорошо слышно
&& видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

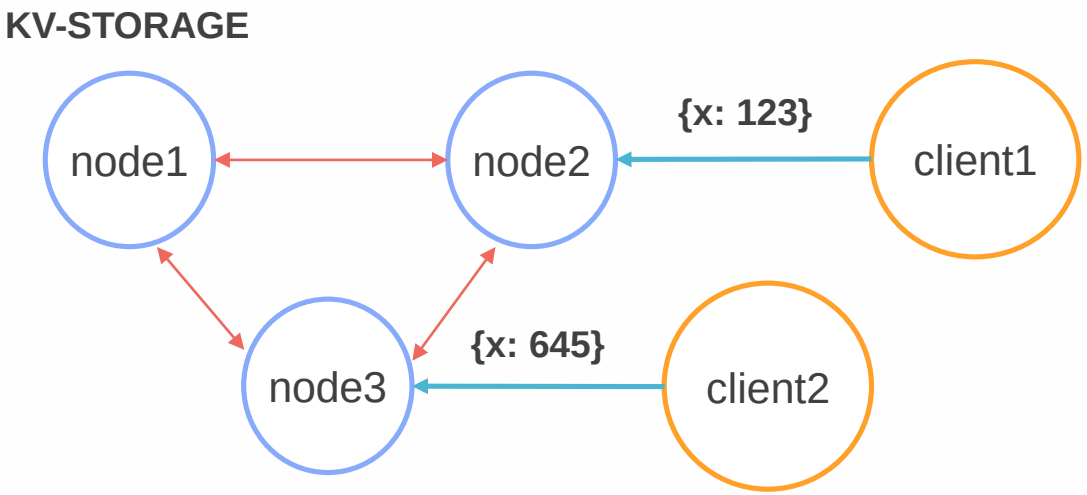
Содержание

- Синхронизация изменений
- Алгоритмы согласования
- Paxos
- Raft
- Zab

01

Синхронизация изменений

Проблема синхронизации

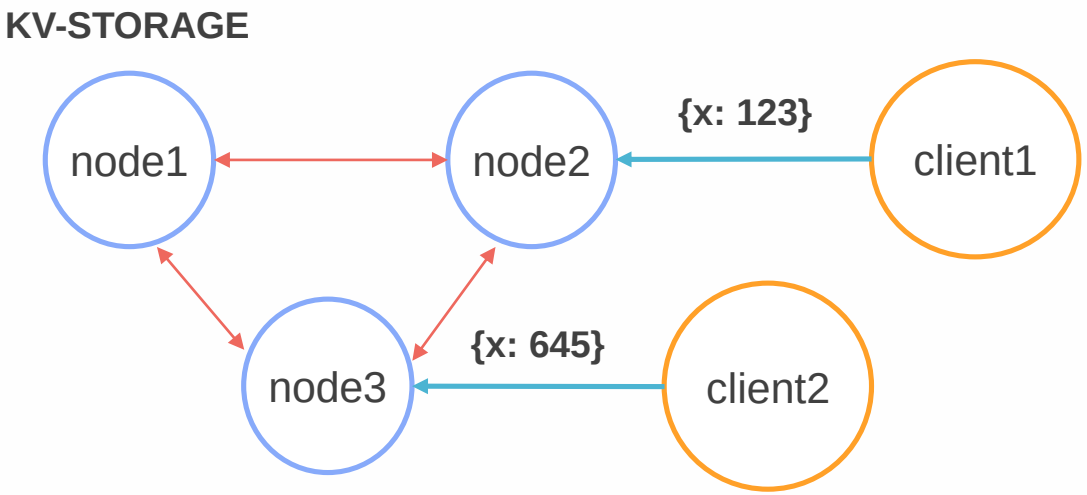


key	value
x	20
y	15

x	?
y	15

- **Две ноды предлагают новое изменение в один момент времени**
- **Изменение является коллизионным**
- **Какое состояние будет в итоге у всех 3 нод**

Проблема синхронизации

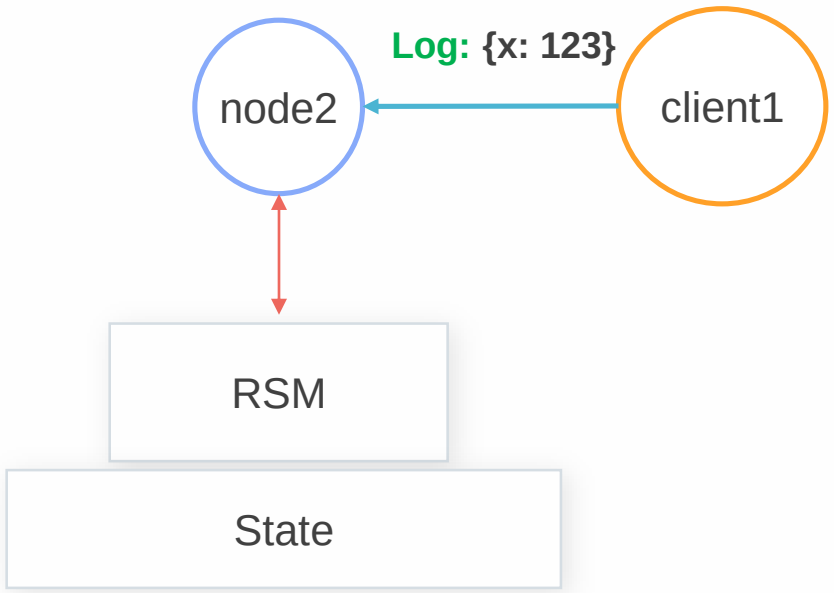


key	value
x	20
y	15

x	?
y	15

- **Согласование изменений.** Кто будет решать?
- **Очередность** (работа State Machine): $x1+x2+x3 \neq x2+x1+x3$

State machine



key	value
x	20
y	15

x	123
y	15

- **RSM** – это механизм (machine) применения новых изменений (replicated changes / logs) на текущее состояние (state) системы.
- **Лог** – это единица изменения состояния (как транзакция). Логи могут храниться и представлять из себя всю историю

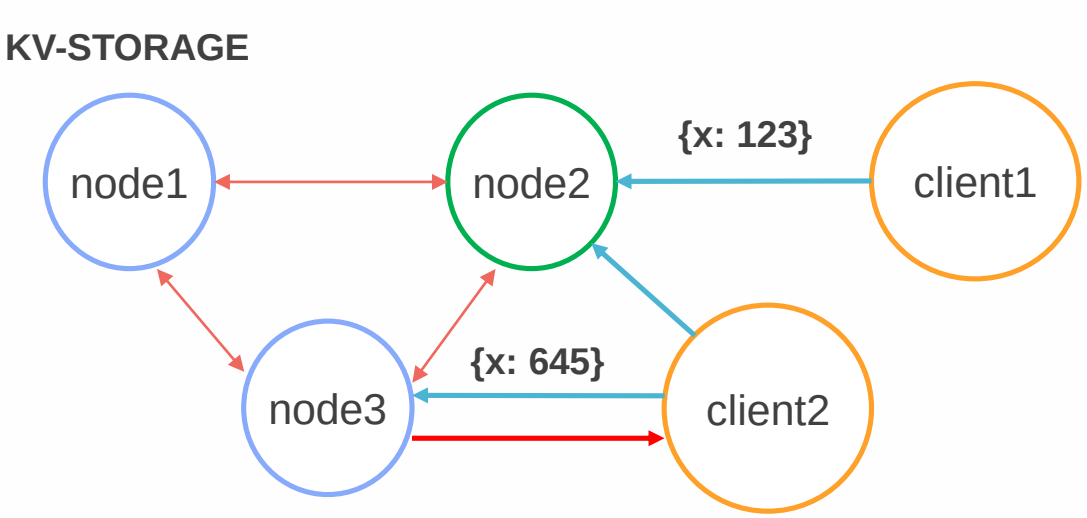
02

Алгоритмы согласования

Алгоритмы согласования

Алгоритмы согласования – это механизм, в распределенной системе, который позволяет участникам, в данной системе, достичь соглашения о конкретной единице информации.

CP (leader based)



key	value
x	20
y	15

x	**645**
y	15

- **Согласование изменений.** Выбирается лидер
- **Все изменения** идут через **лидера**
- Все **другие** ноды являются **follower'ами**

Что можно с ними делать?

- Синхронизация состояния (изменений): базы данных, блокчейн
- Разрешать коллизии (за счет голосования): ZAB (kafka)
- Оркестрация: docker swarm

Основные свойства

- **PC vs PA** – консистентность или доступность
- **Sync vs Async** – таймеры или события
- **Performance** – скорость работы
- **Security** – важна ли безопасность
- **Design** – компоненты системы
- **Development** – ЯП и средства разработки

- Скорость может быть ниже
- Единая точка принятия изменений
- Может быть гарантирована 100% консистентность

Sync vs Async

Синхронная система – вид системы, использующий для синхронизации внешние факторы. Например завязка на время (то есть таймеры).

Асинхронная система – вид системы, который работает на событиях, и не завязан на внешние факторы. Работает быстрее синхронных, но тяжело управлять.

Performance

- Скорость репликации (передача данных)
- Скорость работы RSM (принятия изменений)
- Скорость синхронизации (синхронизация изменений между узлами)

- Где будет работать алгоритм (**открытая ли сеть**)?
- Как будем защищать систему от внешнего воздействия и перехвата трафика?
- BFT

Security -> BFT

Byzantine Fault Tolerance - устойчивость системы к падению / плохому поведению одного / нескольких участников (нод) в сети

Примеры атак:

- Изменение состояние системы (несанкционированное)
- Перезапись логов
- Удержание сети
- Атака 51%

- **Log-less vs Logs** – нужно ли хранить все логи?
- **Компоненты** – из каких частей состоит?
- **RSM**

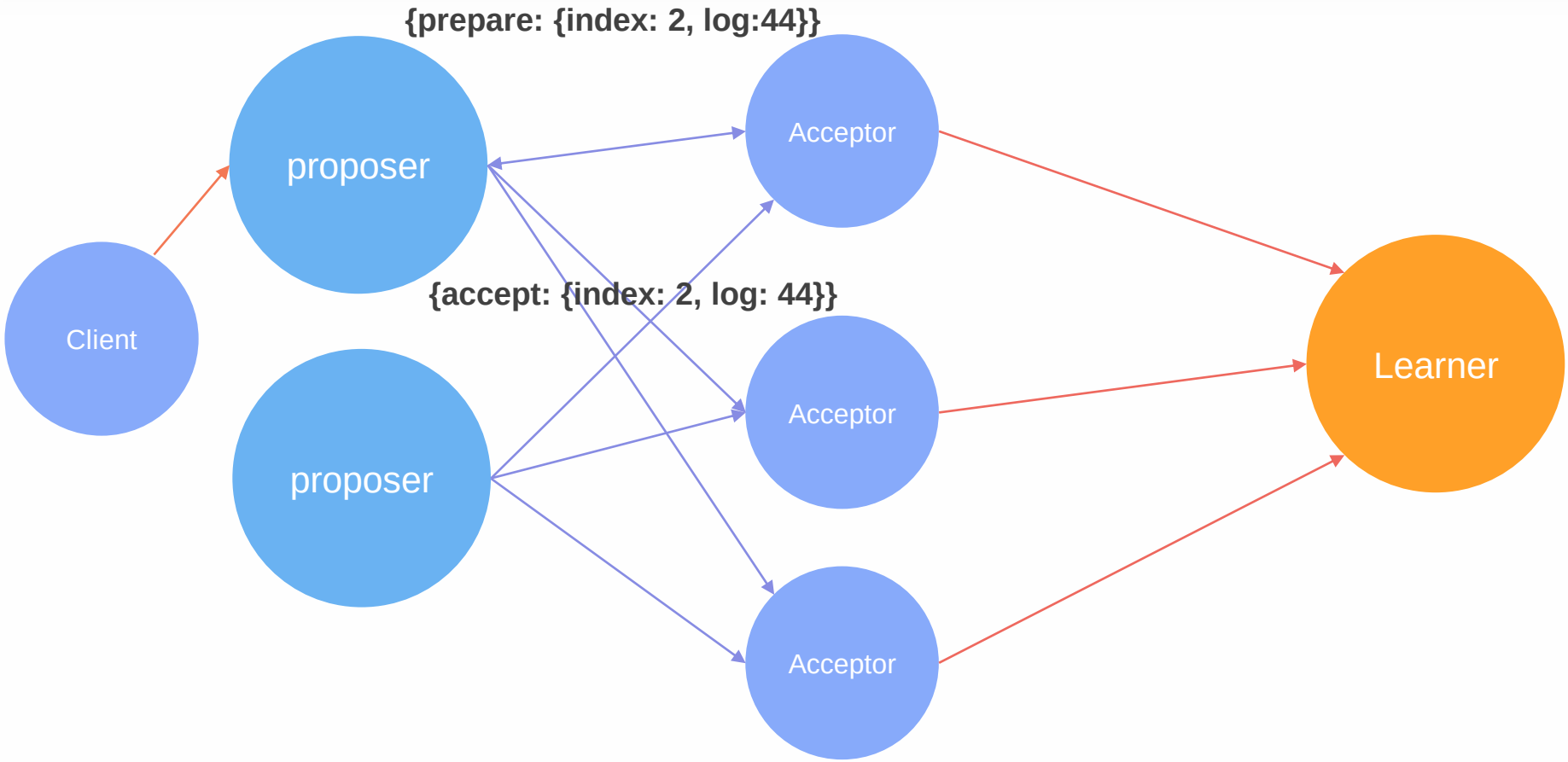
03

Paxos

Свойства

- CP алгоритм (не leader-follower)
- Синхронная система: использует таймеры для определения задержек в сети, и во время ack
- двух-фазный коммит

Принцип работы



Proposer – предлагает изменение, синхронизируется со всеми acceptor'ами

Acceptor – принимает изменение

Leaner – хранит все изменения (то есть как state machine)

Demo

OTUS

http://harry.me/blog/2014/12/27/neat-algorithms-paxos/

Кто использует?

- VMWare (NSX)
- AWS
- Apache Mesos
- Google
- Azure

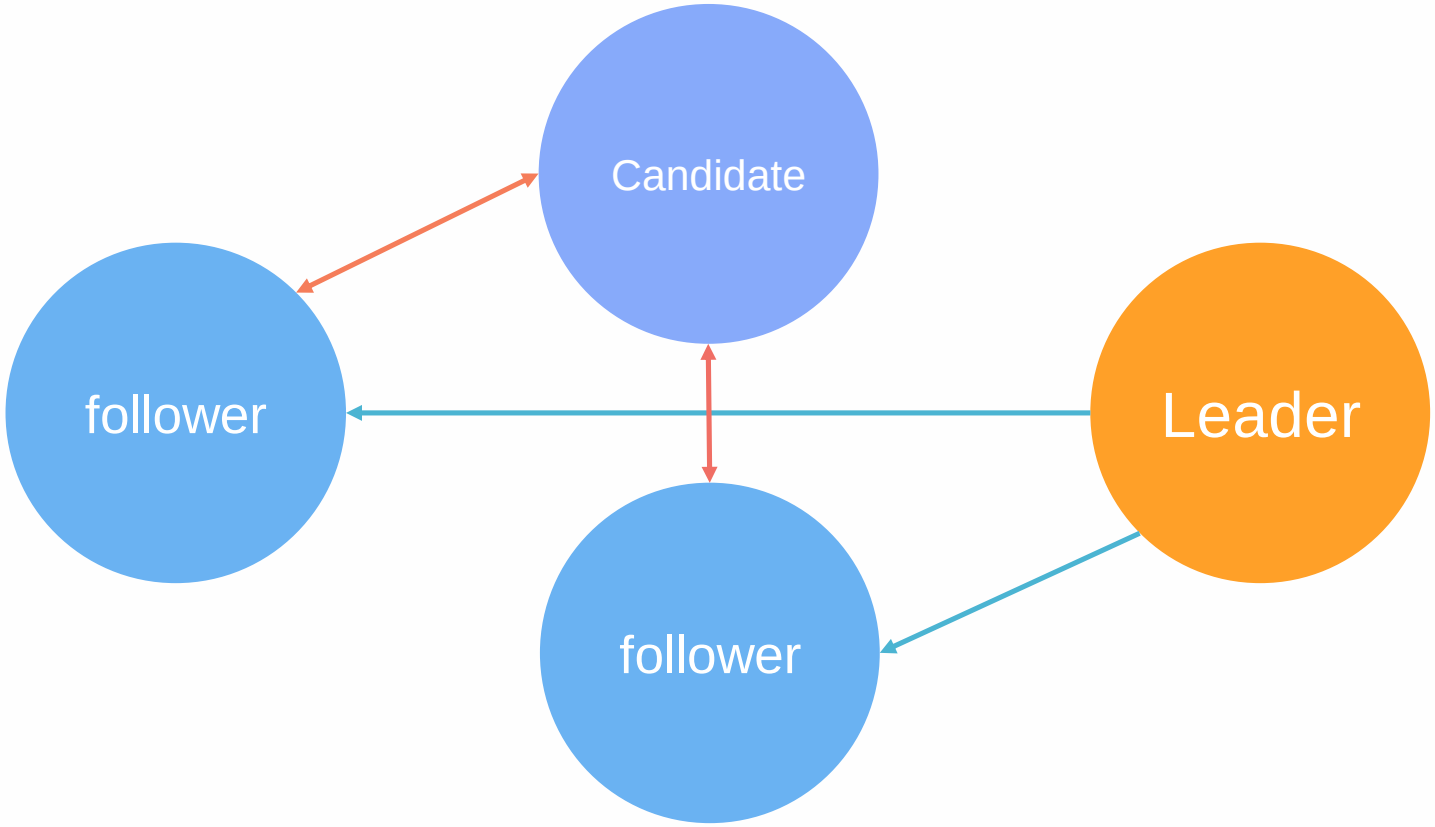
04

Raft

Свойства

- CP алгоритм
- Синхронная система: использует таймеры
- Переработанный концепт paxos

Принцип работы



- Leader – принимает изменения и реплицирует от клиента
- Follower – принимает изменения от leader
- Candidate – должен стать leader или follower

Demo

https://raft.github.io/

Кто использует?

- Consul
- Docker (swarm)
- Apache KUDU
- IBM (Hyperledger sawtooth)
- CockroachDB

05

Zab

- CP алгоритм
- Leader-follower

Определения

- Эпоха leader’а e - число, генерируемое новым leader’ом в начале своего лидирования. Превосходит эпохи предыдущих leader’ов.
- Каждая транзакция имеет номер c, генерируемый leader’ом, начинается с 0 и увеличивается. Используется совместно с эпохой для упорядочивания транзакций.
- У каждого follower’а имеется очередь транзакций, в которой они хранятся в том порядке, в котором они пришли на него (история)

- Читать можно из любой ноды.
- Писать можно в любую ноду, причем follower перенаправит запрос leader’у
- Когда к leader’у приходит изменение, он заводит транзакцию с c и e, начинает фазы prepare и, когда получает ack от кворума, commit. Follower сначала ждет фазы commit’а транзакций с меньшим c.
- При сбое leader’а ноды согласовывают общее состояние, в том числе, решая проблему выбора нового leader’а.
- Жизненный цикл состоит из 4 фаз: leader election, discovery, synchronization, broadcast

- Потенциальный leader получает от follower’ов самую «свежую» последовательность принятых транзакций и устанавливается новая эпоха.

Ресурсы

OTUS

http://book.mixu.net/distsys/single-page.html

http://dimafeng.com/2016/12/04/distributed-systems/

Summary

- Синхронизация изменений
- Алгоритмы согласования
- Paxos
- Raft
- Zab

Ваши вопросы?

Ресурсы

OTUS

http://book.mixu.net/distsys/single-page.html

http://dimafeng.com/2016/12/04/distributed-systems/

Спасибо
за внимание!