数字转型 架构演进

SACC

2019 中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2019

IT168.com ChinaUnix.net ITPUB

2019年10月31-11月2日 | 北京海淀永泰福朋喜来登酒店

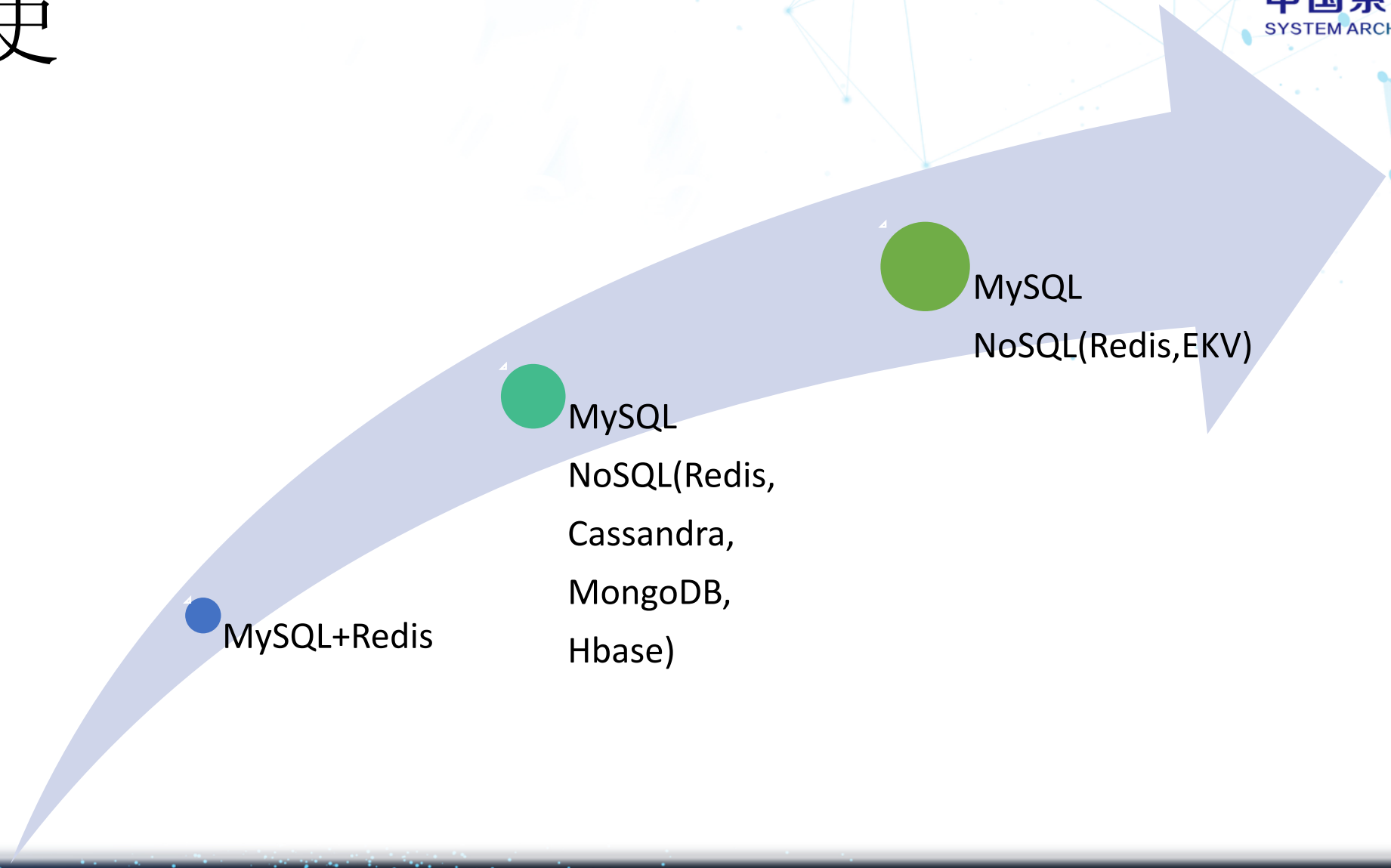
饿了么自研分布式KV数据库的架构与实践

饿了么 核心基础设施部 高级架构师 陈东明

Agenda

- 历史与需求
- 核心特性选择和对比
- 架构简介
- 架构设计中的权衡
- 未来的规划

历史


MySQL+Redis
MySQL
NoSQL(Redis,
Cassandra,
MongoDB,
Hbase)
MySQL
NoSQL(Redis,EKV)

需求

- 痛点
  - 海量的、持久化的数据存储
  - 对于我们自己
    - 统一NoSQL使用、统一运营、降低成本、适用多种场景
  - 对于我们的使用者
    - 简单易用
- 其他需求点
  - 数据可靠、服务稳定、高性能、低维护成本

Agenda

- 历史与需求
- 核心特性选择和对比
- 架构简介
- 架构设计中的权衡
- 未来的规划

小调查

- 什么是强一致性？
- 你认为数据库需要强一致性吗？

What is EKV

我们EKV是一个强一致性数据库

选择强一致 – 未来的方向

- MySQL
  - 从v5.7开始，支持Group Replication，采用Paxos
- MongoDB
  - 从v3.4开始，支持类Raft复制协议
- Hbase
  - 类似于BigTable/GFS *
- NewSQL
  - Spanner,TiDB,CockroachDB等数据库采用Paxos或Raft

➢ 除元数据存储以外，越来越多的数据库采用类Paxos算法
➢ 越来越多的数据库采用强一致性

* 《从 GFS 失败的架构设计来看一致性的重要性》，陈东明，https://mp.weixin.qq.com/s/GuJ6VqZJy3ONaVOWvQT9kg

对比

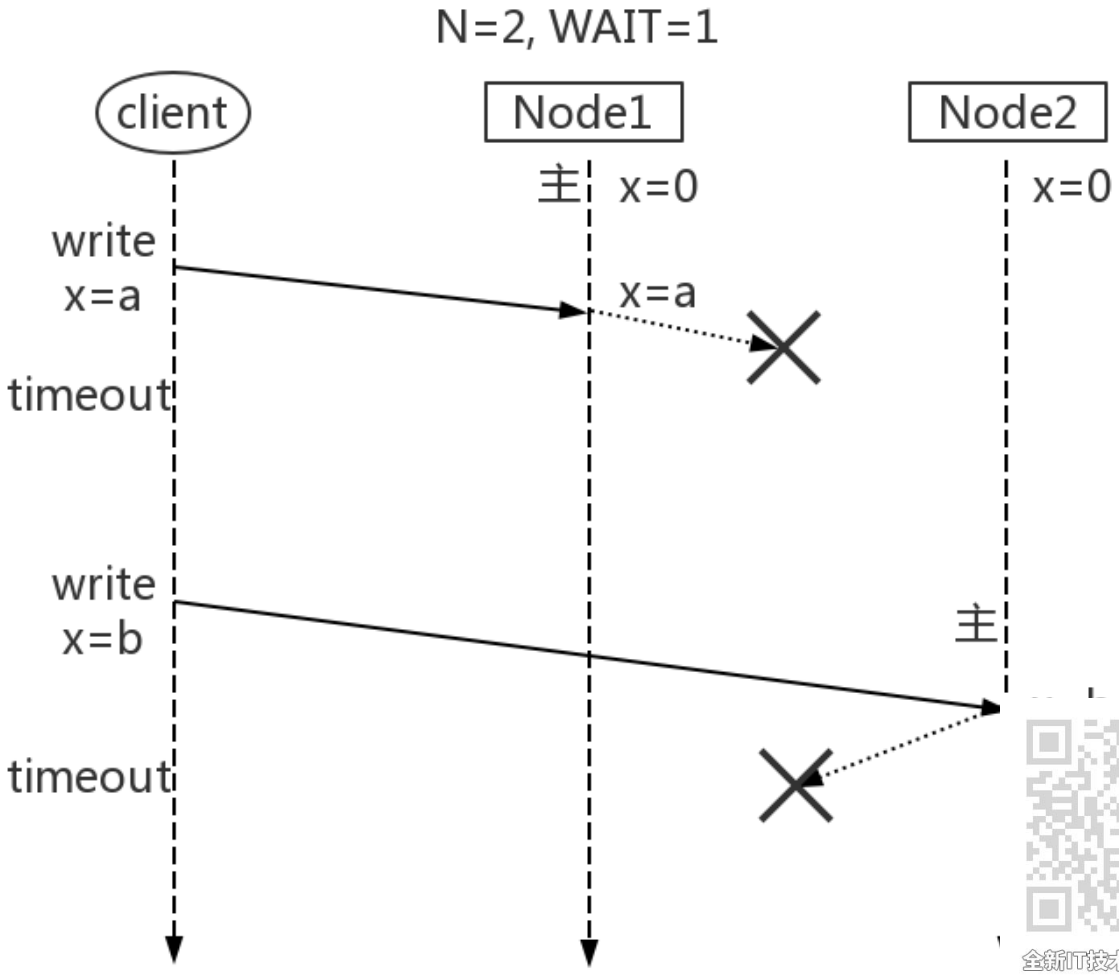
- 对比三个典型系统
  - MySQL
  - Redis
  - Cassandra

MySQL

- MySQL
  - 主备异步复制
  - 非强一致性的（在出现故障时）
  - Lost data
- 使用建议
  - Strong DBA for failover
  - Carefully programming with slave node
- 架构经验
  - ***异步复制不能保证强一致***

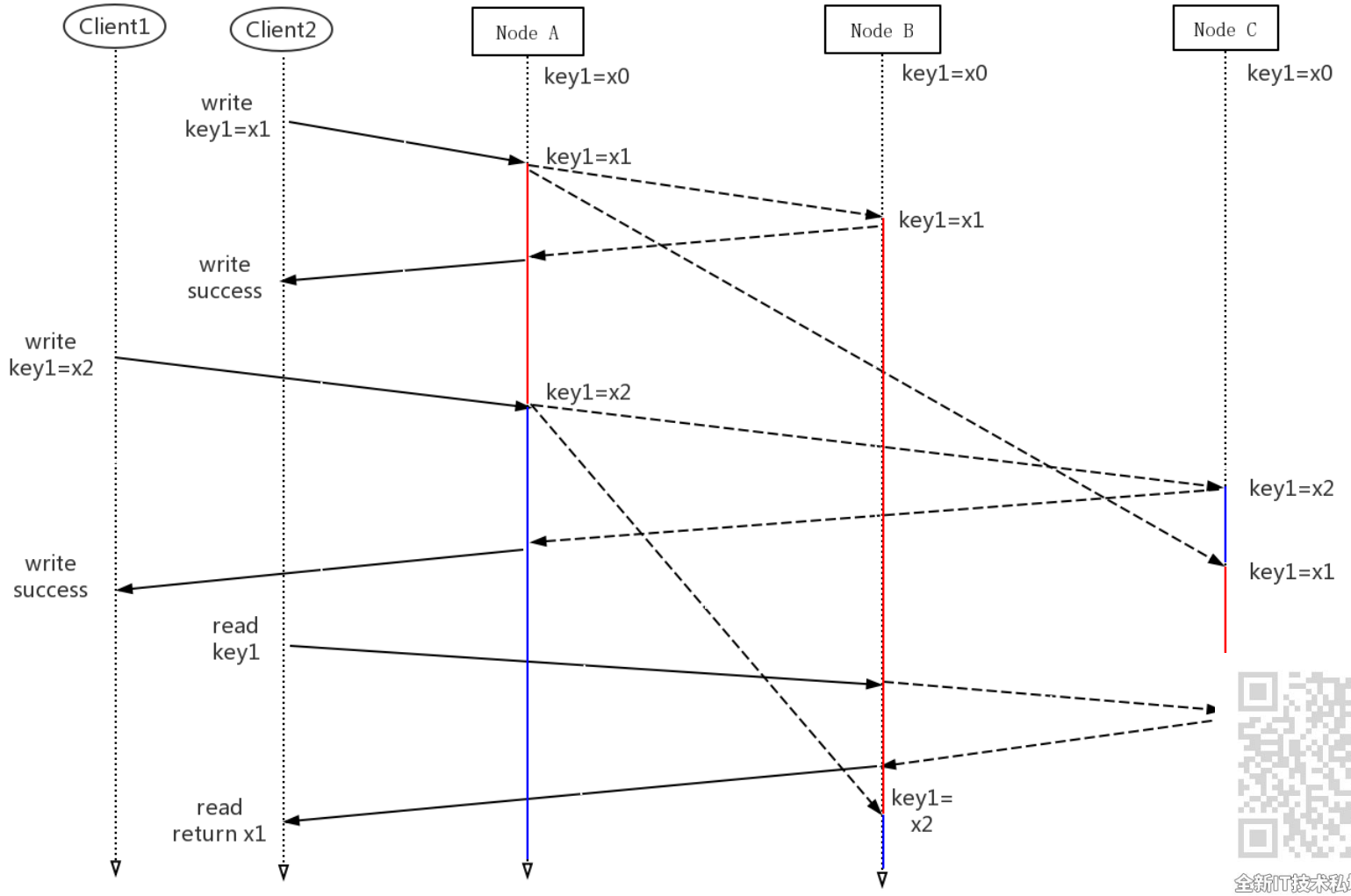
Redis

- Redis
  - 异步复制
  - WAIT指令，同步复制
  - 副本不一致（即便使用WAIT）
  - 非原子
- 使用建议
  - WAIT不应该成为一个对外暴露的API
  - Carefully programming with WAIT
- 架构经验
  - ***同步复制不能保证强一致***



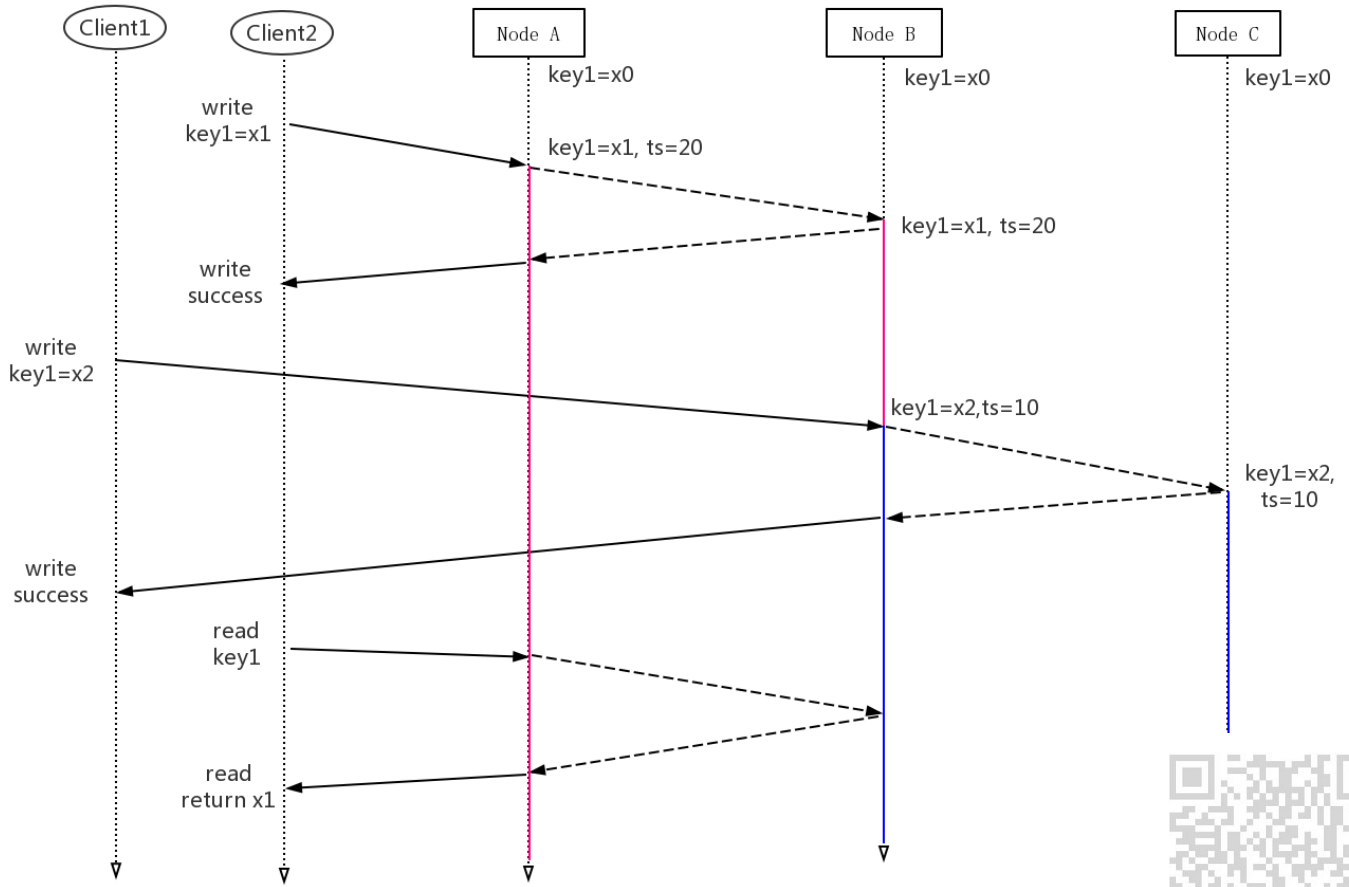
Cassandra/Dynamo

- WRN策略
- Quorum(W+R>N)
  - 副本不一致
  - Stale Read
  - 并发冲突
- 如何解决副本不一致？
  - 再读一次（试试运气）
  - Read Repair
  - Replica synchronization



Cassandra/Dynamo

- 冲突解决
  - Last Write Win
  - Clock skew（或者叫时钟不同步）



Cassandra/Dynamo-- 最终”正确”一致性

- Cassandra/Dynamo
  - 最终一致性（任何情况下，即便W+R>N）
- 使用建议
  - 放在无需数据正确性的场景
  - 或者做好补偿措施
- 架构经验
  - ***W+R>N不能保证强一致性***
    - 在并发或者故障时
  - ***最终一致性≠保证最终正确的一致性***

使用强一致的原因

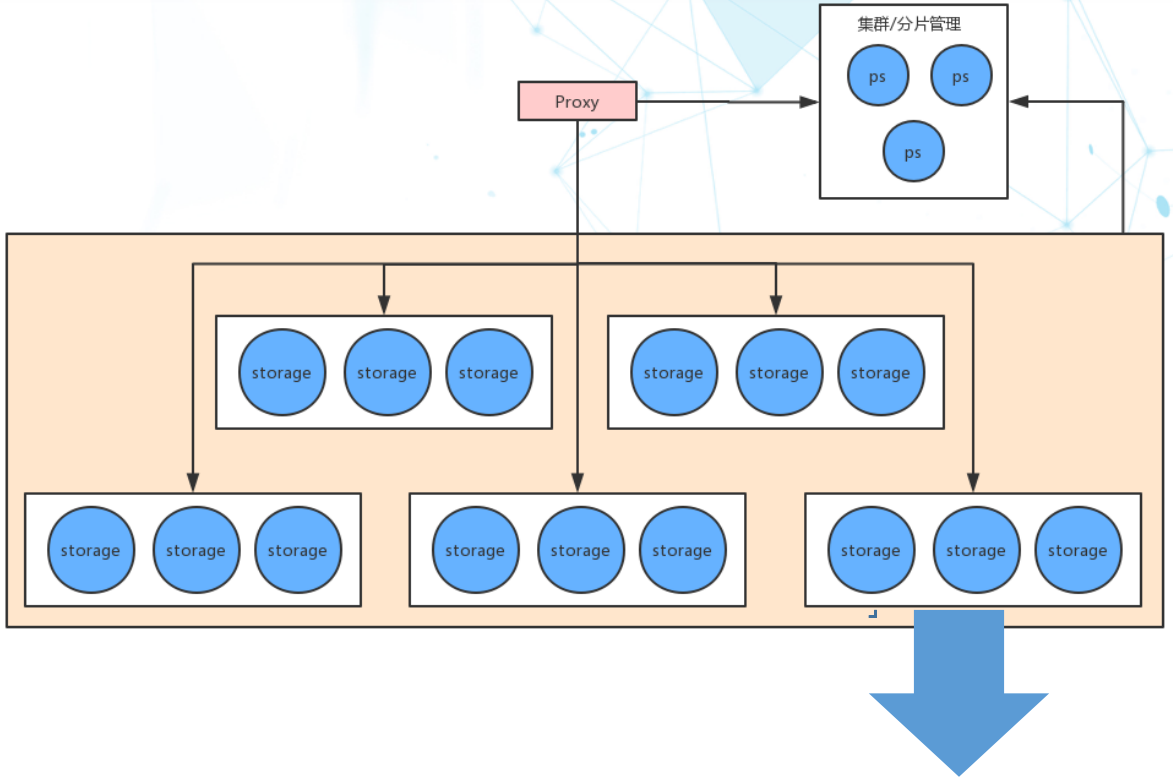
- 数据正确性
- 一致性本质上是并发问题和故障处理问题
- 低开发成本
  - 易懂
  - 易用
  - 易维护
- 让开发这个事更方便，解放开发者

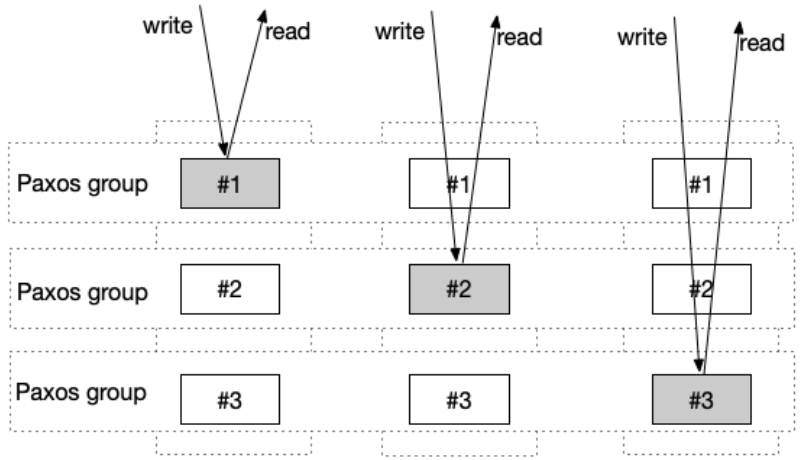
Agenda

- 历史与需求
- 核心特性选择和对比
- 架构简介
- 架构设计中的权衡
- 未来的规划

架构

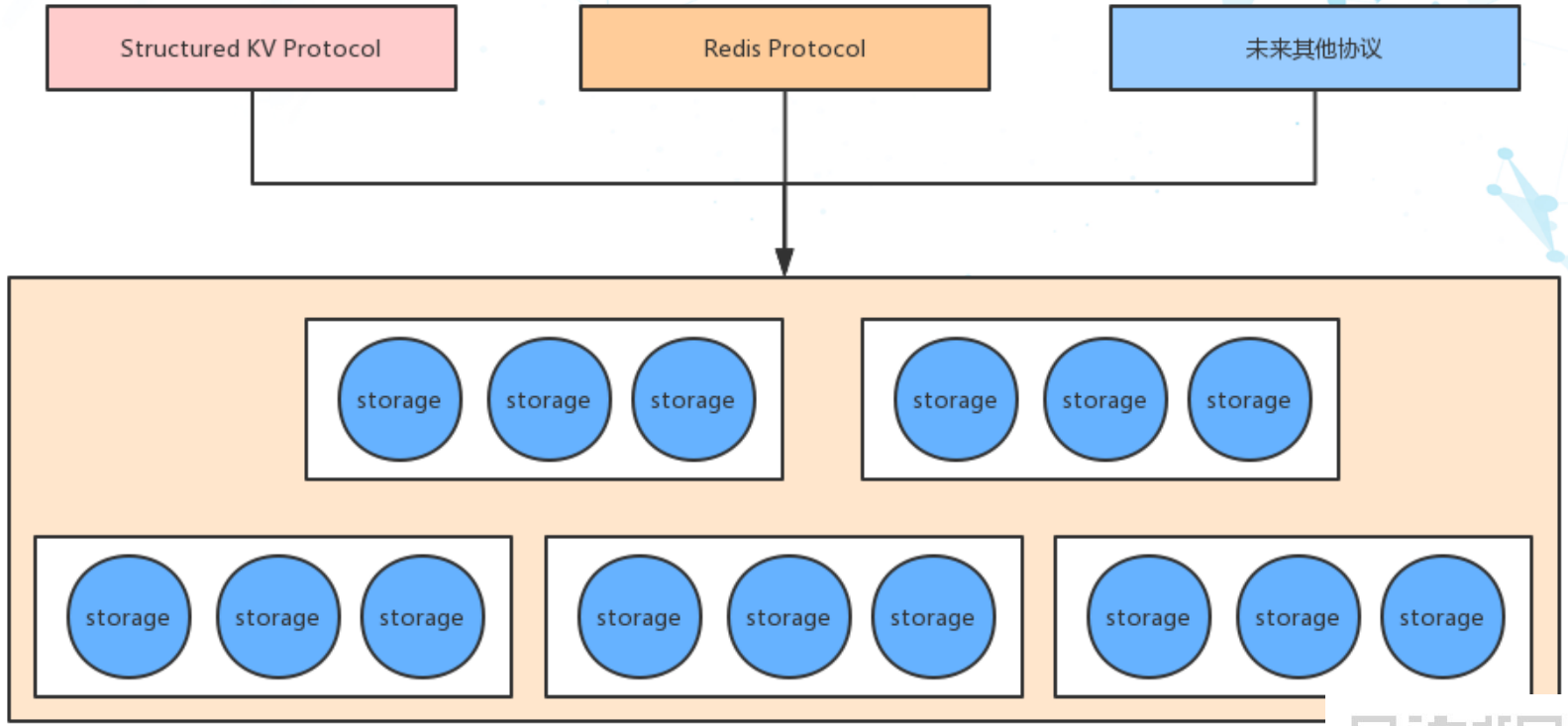
- 分片(Sharding)
  - ***海量数据存储***
- Paxos
  - 多副本（***高可靠***）
  - 大多数原则
  - Fault-tolerant（***高可用***）
- ***强一致性***





其他功能

- 多数据模型
  - Structured KV模型
    - Integer
    - String
    - Sorted Hash(稀疏大表)
  - Redis模型
- 异地多活
- 水平扩容



Agenda

- 历史与需求
- 核心特性选择和对比
- 架构简介
- 架构设计中的权衡
- 未来的规划

是否有这样的疑虑

- 强一致性的系统可用性不好？
- 强一致性的系统性能不好？

强一致vs.可用性

- CAP定理
  - Availability，指的是100%绝对可用（任何时间，哪怕只剩一个节点）
- CAP定理不适用Paxos
  - Paxos容忍少数宕机，虽然没有达到完全可用，但仍然很高的可用性
- 需要绝对可用的架构设计吗？不,需要是***实际的高可用***, ***Effectively CA*** *，**
  - Google的强一致系统达到99.99958%的实际可用性
  - 网络分区并是不可用的主要原因，***架构选择可用性≠实际获得高可用***

* Spanner, TrueTime & The CAP Theorem， Eric Brewer， https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45855.pdf
** NoSQL 数据库不应该放弃 Consistency，陈东明， https://www.infoq.cn/article/rhzs0KI2G*Y2r9PMdeNv

强一致vs.性能

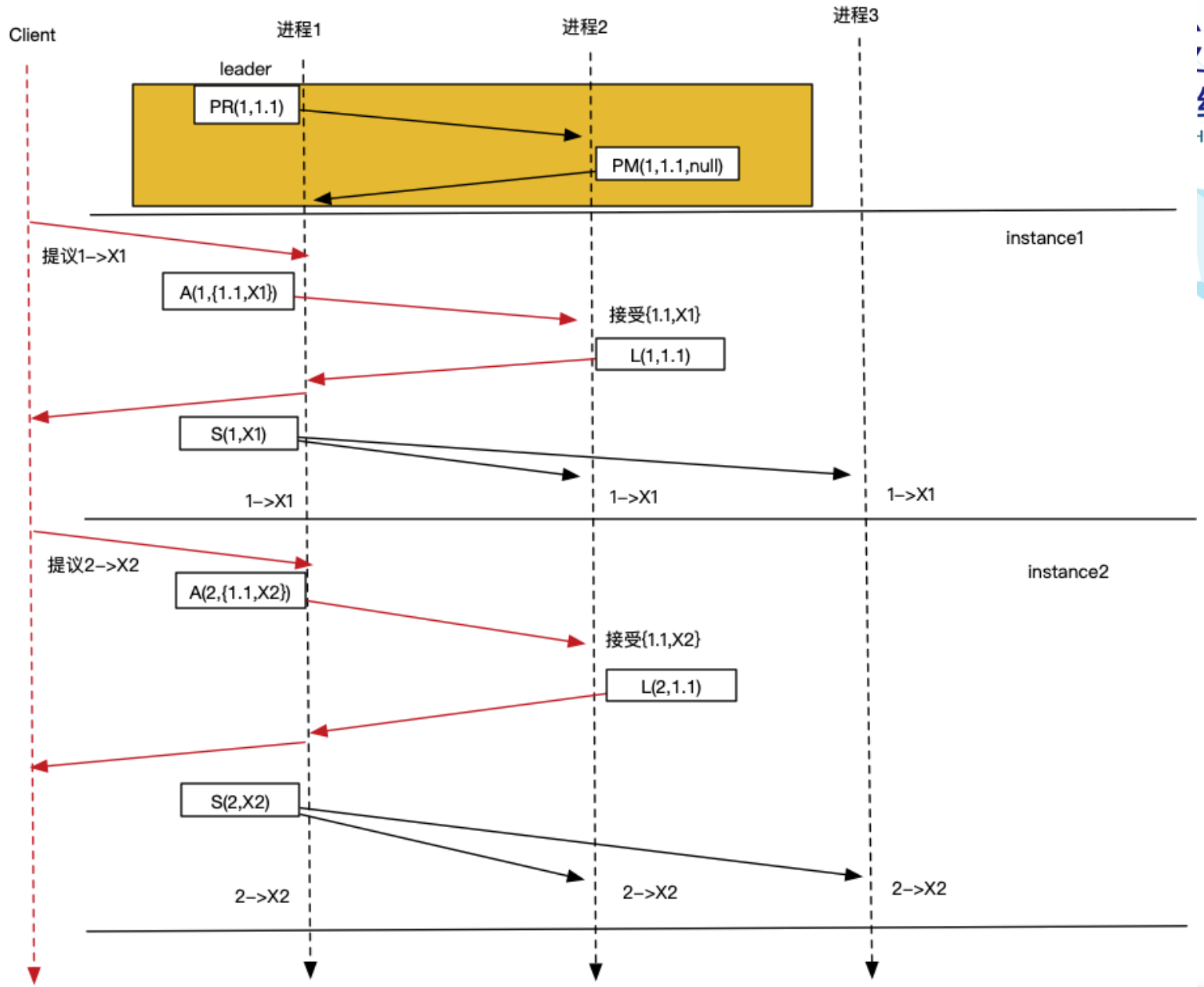
- What is performance?
  - Throughput
    - 采用数据分片，在大多数场景下Throughput没有影响
  - Latency
    - 复制过程的Network Round Trip

* 3个物理机节点，64core，2.8G，256G内存，1块1T的SSD硬盘，库中已有数据500G，200个sharding分片，ValueLength=300B

Paxos

latency

latency



强一致vs.性能

- Round Trip（以三个节点为例）
  - Write Network RT：异步复制 < Paxos = (W+R>N) < 同步复制

异步复制	Paxos	(W+R>N)	同步复制
0RT	1RT	1RT	1RT
0	最快的	最快的	全部

  - Read Network RT： 异步复制 = Paxos = 同步复制 < (W+R>N)

异步复制	Paxos	同步复制	(W+R>N)
0RT	0RT	0RT	1RT
0	0	0	最快的

- 我们的实践， Write latency <= 5ms， Read latency <= 1ms
  (2w write QPS， 10w read QPS) *
- ***高性能***

* 3个物理机节点，64core，2.8G，256G内存，1块1T的SSD硬盘，库中已有数据5ððT，200个sharding分片，ValueLength=300B

Agenda

- 历史与需求
- 核心特性选择和对比
- 架构简介
- 架构设计中的权衡
- 未来的规划

规划

- 更丰富的数据模型
  - 文档数据模型
  - 图数据模型
- 优化
- 分布式事务

总结（1）

系统架构设计的经验：

- 异步复制、同步复制、 W+R>N ≠ 强一致
- 最终一致性 ≠ 保证最终正确的一致性
- 重新考虑CAP的二选一：
  - 架构上选择可用性 ≠ 实际上获得高可用
  - 架构上选择一致性 ≠ 实际上可用性低，仍然可以实际高可用
- 强一致性 ≠ 低性能

总结（2）

- 构建了一个自研的、不同于现有NoSQL数据库的：
  - EKV是多数据模型、强一致的key-value数据库
  - EKV具有高可用、高可靠性、海量存储、高性能的能力

Slogan

Q&A

个人微信号

公众号

Thanks