

可靠性、可扩展性与可维护性

DDIA 读书分享

数据系统的模型、存储与分布式取舍

wxquare · 原分享 2023-12-20

公开版整理 2026-10-12

DDIA：一本讨论数据系统取舍的书

Designing Data-Intensive Applications

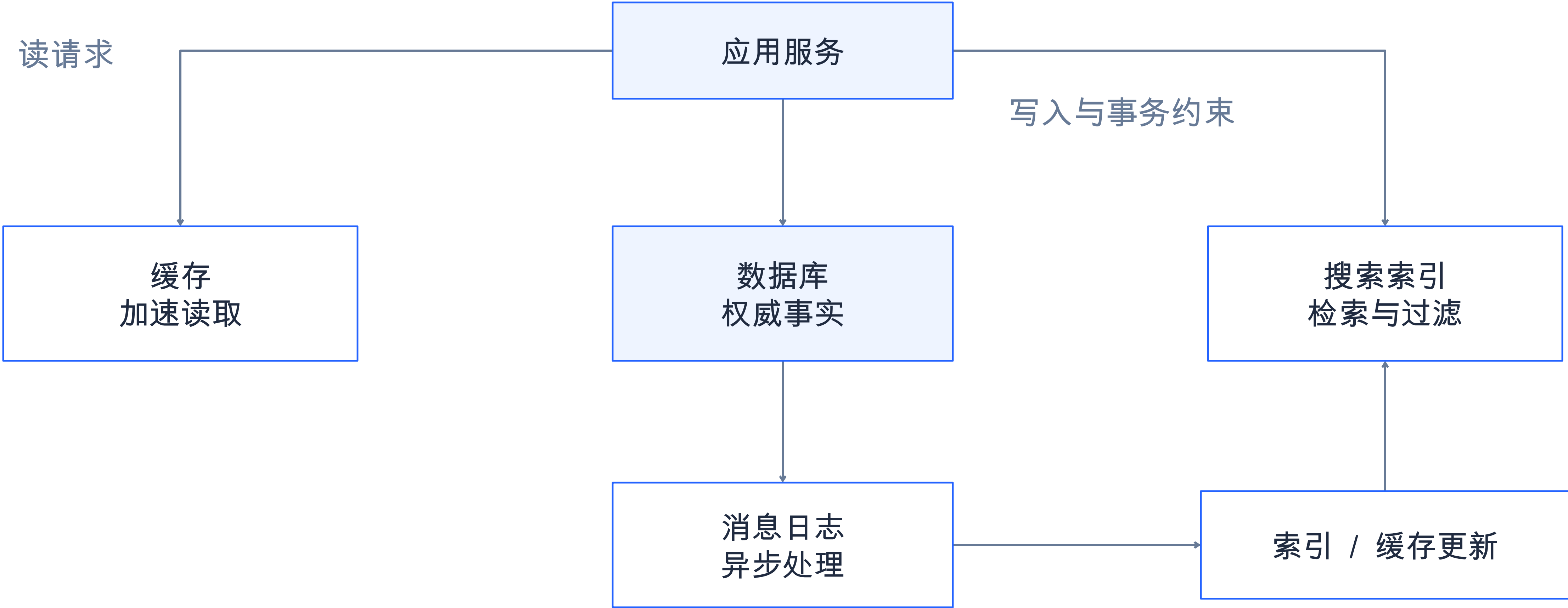
Martin Kleppmann · 第一版，2017

- 01 可靠性：组件出错后，系统仍按约定提供服务。
- 02 可扩展性：描述负载增长后，吞吐与延迟如何变化。
- 03 可维护性：让运行、理解和演进系统的成本可控。

阅读顺序

数据模型与存储引擎 / 事务、分片与复制 / 组件与数据处理

数据系统：权威事实与派生数据



复制延迟与更新失败会造成短暂差异。设计要明确允许陈旧多久、如何重建及核对。

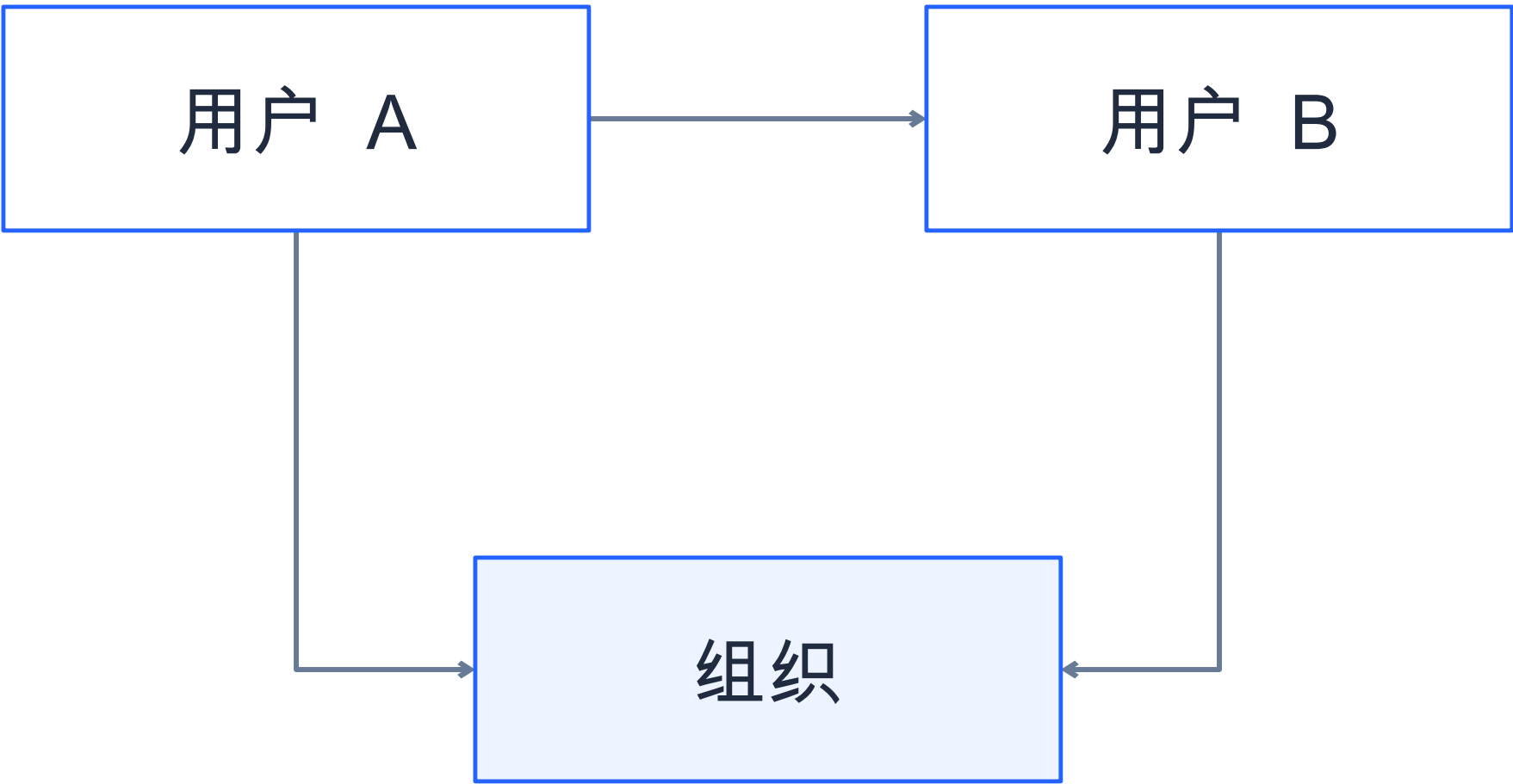
数据模型： 关系与文档的选择条件

维度	关系模型	文档模型
数据关系	关联查询、多对多关系	聚合内嵌套、一对多关系
约束与演进	约束集中，迁移需规划	结构灵活，仍需兼容策略
访问局部性	关联与索引决定读取成本	整份文档读取更自然
事务边界	依具体引擎与隔离级别	可支持多文档事务，需核对代价

事务、Join 与 schema 能力取决于具体产品和版本，不能仅凭模型类别判断。

图模型、列式存储与宽列模型

图模型：关系遍历



关注关系、路径与邻居查询
需评估遍历范围和关系更新成本

列式存储：按列扫描



聚合只读需要的列，利于压缩与扫描。
逐行、小范围更新需评估写入成本。

宽列模型与列式分析存储是不同概念。
数据稀疏或量大，不足以单独决定选型。

存储引擎：LSM Tree 与 B+ Tree

维度	LSM Tree	B+ Tree
写入路径	内存缓冲、日志、落盘与合并	按页更新，日志保护修改
读取路径	查询内存和多个有序文件	按树定位页，适合范围访问
后台成本	Compaction、写放大和尾延迟	页分裂、碎片与随机 I/O
选择依据	写入压力、压缩与合并预算	读写比例、索引与缓存命中

“LSM 写快、B+ Tree 读快”是粗略直觉。最终结论需要真实数据与负载验证。

ACID 、 CAP 与一致性的不同边界

概念	讨论的问题	评审时需要明确
ACID	事务原子性、约束、隔离、持久性	业务不变量与事务边界
CAP	网络分区时，线性一致性与可用性	哪些操作拒绝、等待或返回旧值
BASE	对基本可用与最终收敛的概括	收敛条件、修复路径与时间界限
一致性模型	读写结果与顺序的可观察保证	会话、对象和跨对象的保证

CAP 的可用性是理论定义。不能用产品名称直接推导整个业务系统属于 AP 或 CP 。

事务隔离：读可见性与并发写入

并发异常

- 01 脏读、不可重复读与幻读。
- 02 丢失更新：读旧值后覆盖他人修改。
- 03 写偏斜：各自修改不同记录，却共同破坏跨记录约束。

实现选择

单条原子更新

```
UPDATE counters  
SET value = value + 1  
WHERE id = 42;
```

加锁或版本条件更新

```
SELECT ... FOR UPDATE  
UPDATE ... WHERE version = v  
检查影响行数，并处理冲突重试。
```

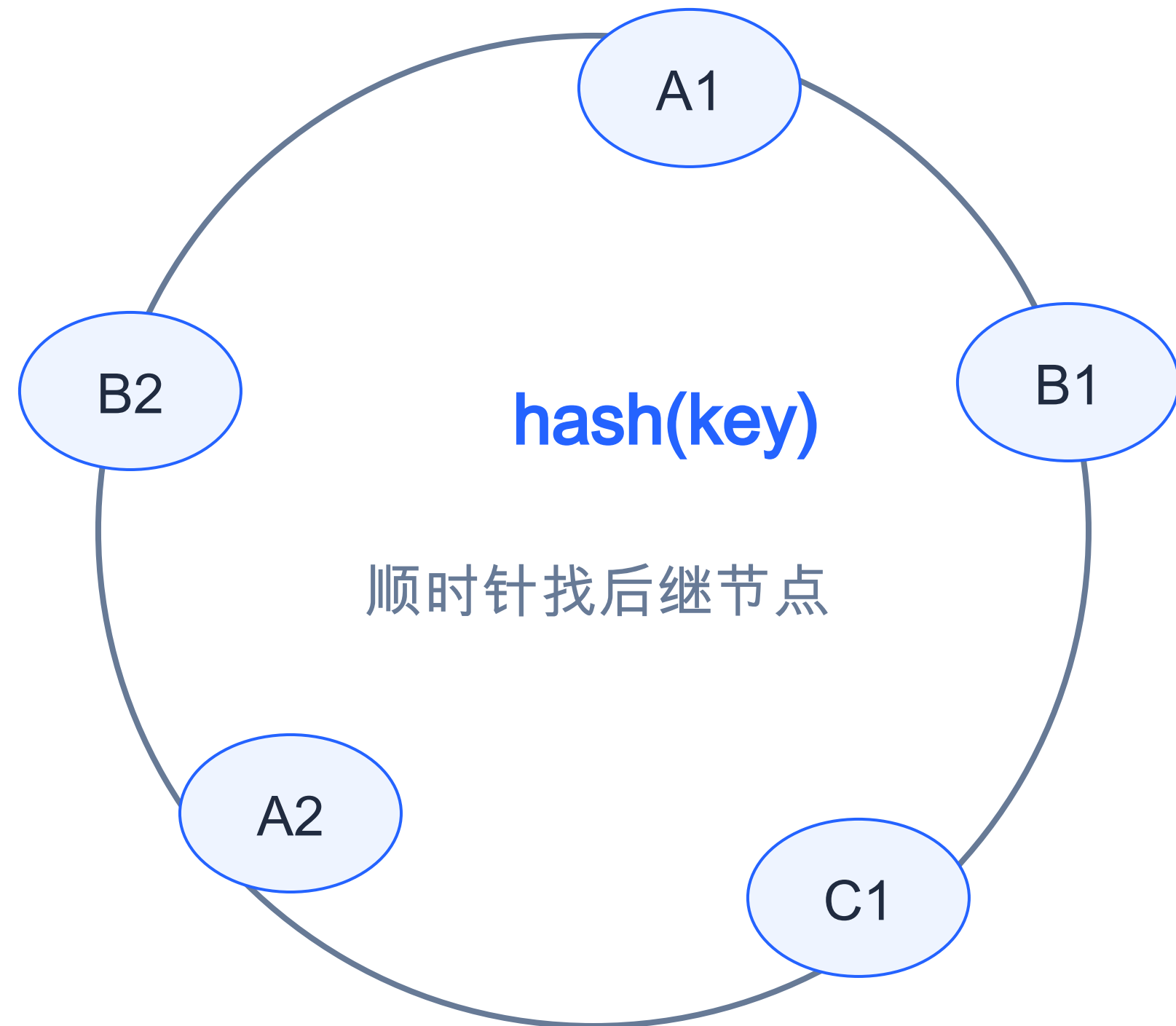
快照隔离未必防止写偏斜。跨记录不变量需要约束、显式锁或可串行化事务。

数据分片：路由与再平衡

策略	有利的访问模式	扩缩容与热点风险
哈希取模	按键点查，路由简单	节点数变化可能导致大规模迁移
范围分片	范围扫描与按时间读取	可拆分区间，需处理尾部热点
一致性哈希	按键点查、节点变化频繁	限制迁移范围，需虚拟节点与迁移协议
映射表 / 逻辑槽	路由与物理节点解耦	维护映射版本、双写与切换流程

分片键也决定事务范围与查询扇出。路由计算不能替代数据复制和迁移。

一致性哈希：虚拟节点与迁移范围



- 01 虚拟节点映射到物理节点，可按容量配置节点权重。
- 02 新增节点接管部分键区间，迁移仍需复制、核对与路由切换。
- 03 虚拟节点改善分布，单个热点键仍需单独处理。

节点删除时，原区间转移到后继节点。复制策略需要另外定义。

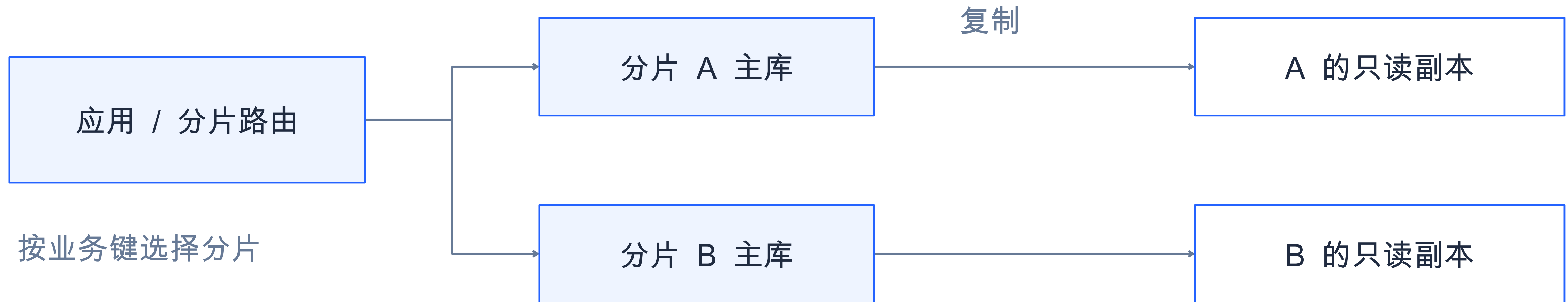
副本复制：单主、多主与无主

模式	常见用途	需要承担的责任
单领导者	集中写入，副本分担读取	复制延迟、故障切换与旧主隔离
多领导者	多地域写入、离线协作	冲突检测、合并规则与因果关系
无领导者	Dynamo 风格的副本读写	版本比较、读修复与反熵

Quorum：读集合与写集合需要交叠

即使 $R + W > N$ ，仍要核对 sloppy quorum、并发写入、故障与版本处理。

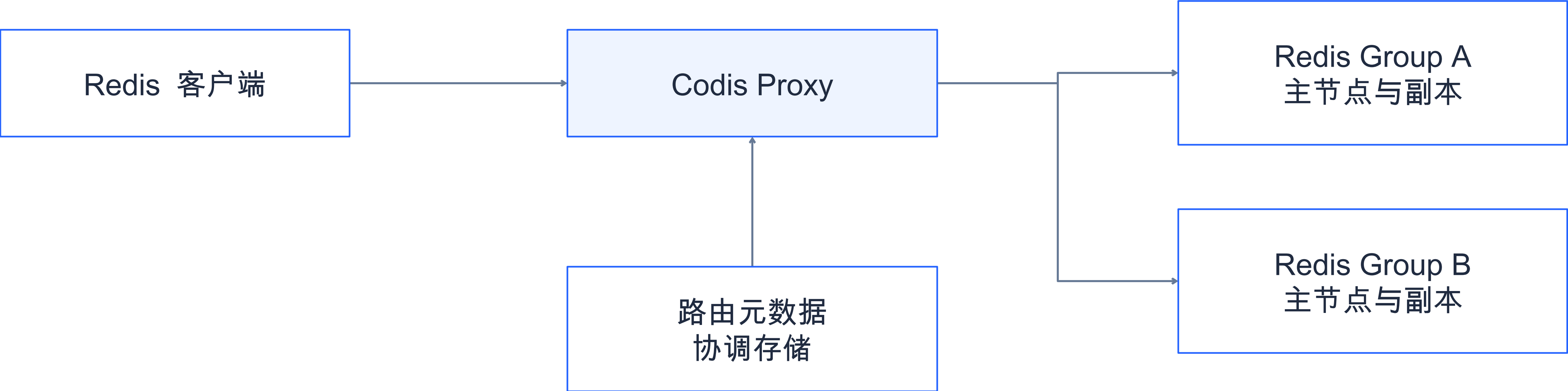
MySQL：分片、复制与故障切换



- 01 增加副本主要提升可接受陈旧数据的读取能力。
- 02 异步复制可能丢失尚未复制的已确认写入。
- 03 切换需隔离旧主，核对数据，并处理结果未知与重试。

半同步确认不等于副本已执行事务。备份、复制与故障恢复需要分别设计。

Codis：代理路由与逻辑槽

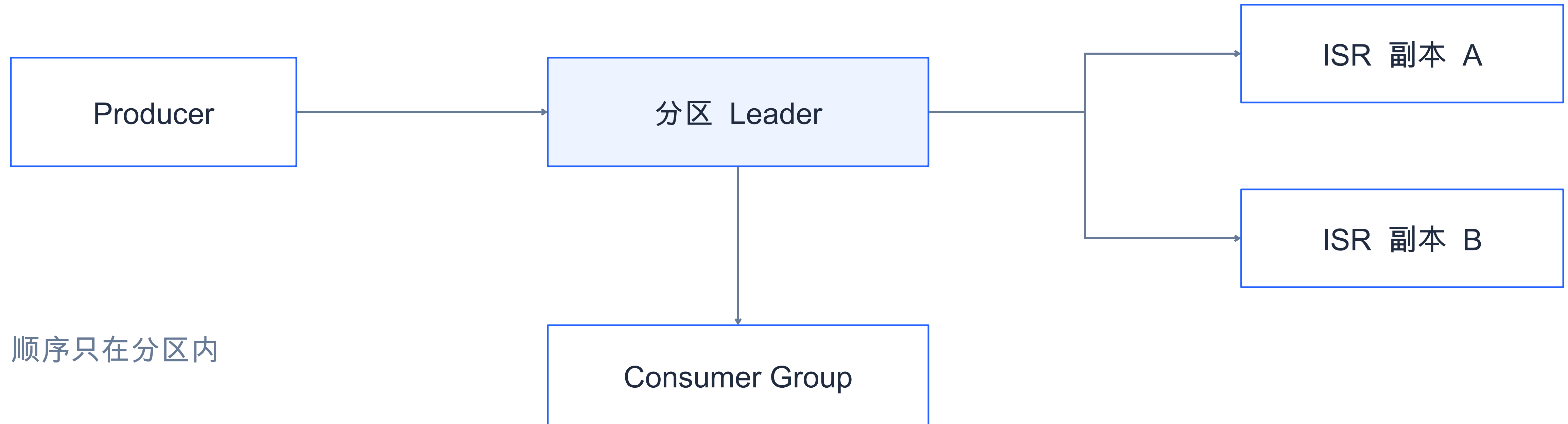


逻辑槽映射到 Group

应用与物理节点解耦，扩容时迁移槽内数据。
代理高可用、元数据一致性和复制可靠性需要分别评估。

这是历史架构学习案例。生产选型应比较现有方案的维护状态与操作成本。

Kafka：分区顺序与写入确认

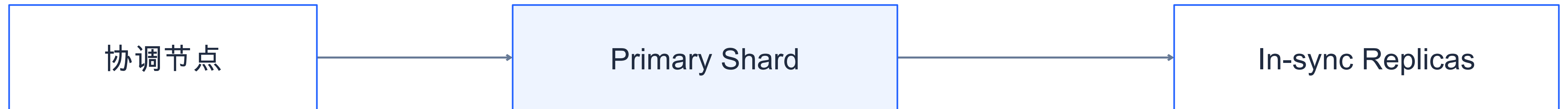


顺序只在分区内

- 01 acks=all 配合 min.insync.replicas 决定写入接受条件。
- 02 Producer 幂等性不能替代外部数据库或支付操作的幂等。
- 03 监控 lag、最老消息年龄与处理速率，评估 rebalance 的影响。

写入保证还依赖副本存活、选主与错误处理。现代 Kafka 的元数据管理使用 KRaft。

Elasticsearch：写入复制与搜索可见性

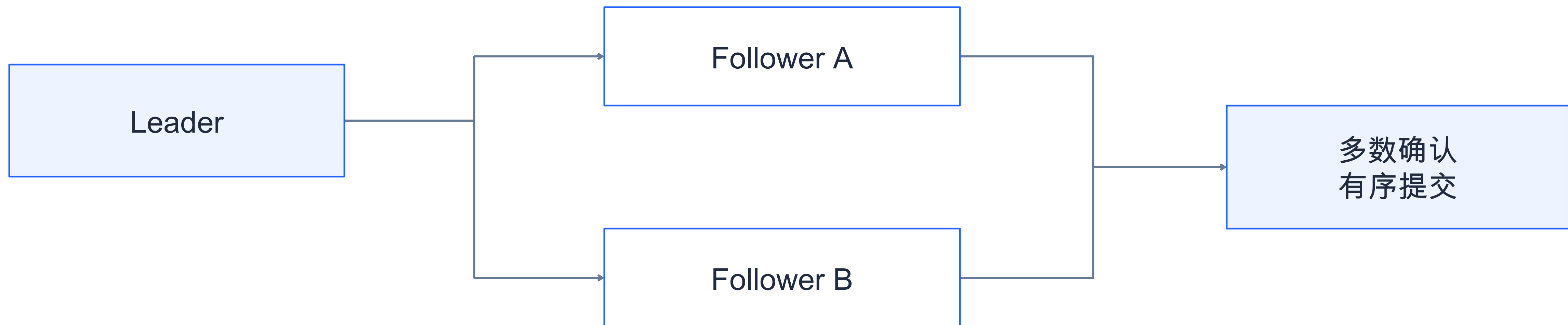


写入在主分片执行，再复制到 in-sync 副本。

- 01 写入成功与搜索可见性分开，refresh 才让新数据可检索。
- 02 Lucene 使用倒排索引与不可变 segment，后台执行 merge。
- 03 分片调整可通过 split、shrink 或 reindex，分别有条件与成本。

Elasticsearch 不使用 Kafka 的 ISR 术语。分片分配、复制与集群选主各有职责。

ZooKeeper：有序写入与读取边界



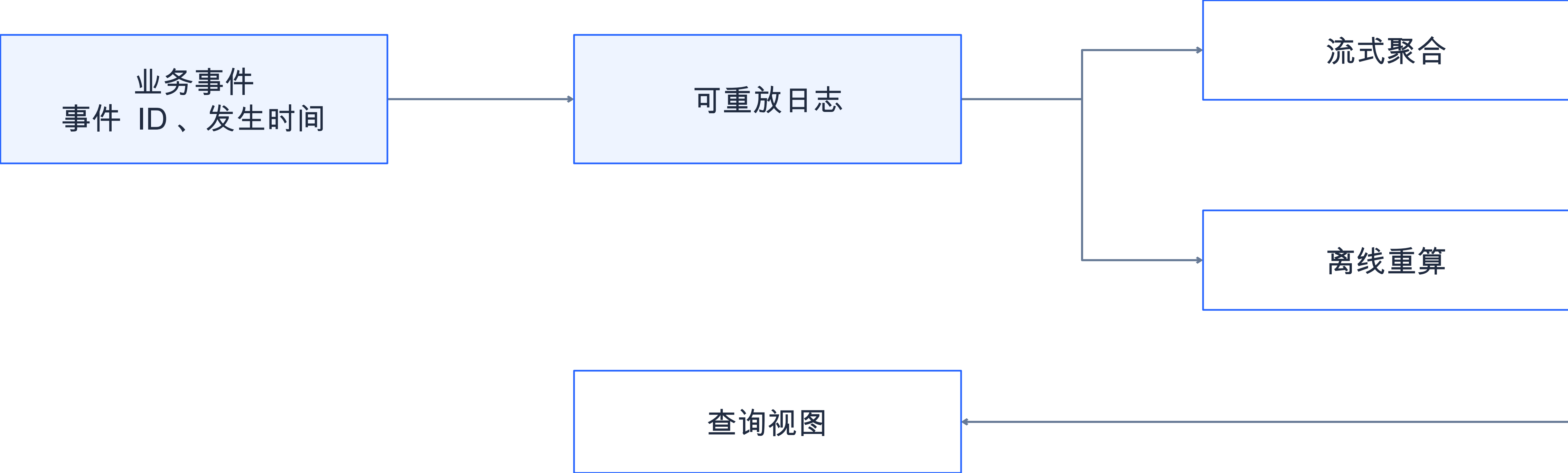
Zab 将状态变更组织为有序事务日志。

Leader 故障后，需要选举与恢复同步，形成一致的历史。

普通读可能返回旧数据。跨客户端观察需结合 sync、版本检查和会话语义。

“强一致性”需要说明具体操作与保证。Zab 的原子广播与两阶段提交不是同一概念。

批处理与流处理：事件统计教学案例



分钟窗口指标按事件时间聚合，迟到数据需要更正策略。

去重依赖稳定事件 ID。重放与重算时使用版本或幂等覆盖，
监控输入量、处理延迟、最老事件年龄和核对差异。

阅读资源与设计问题

设计评审时可以反复问

- 01 哪个组件保存权威事实，哪些数据可以重建？
- 02 写入成功确认了什么，超时后怎样判断结果？
- 03 副本延迟、节点故障与重放会怎样改变业务状态？

原书与阅读资源

dataintensive.net
github.com/Vonng/ddia
github.com/ept/ddia-references

配套文章与课件
wxquare.github.io