KINGBASE | 金仓社区

数据库高可用架构演进和实战应用

分享嘉宾：李翔

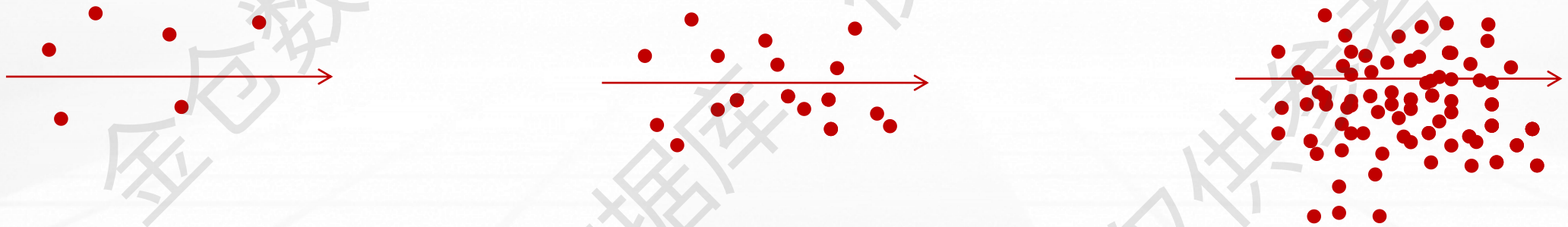
King大咖面对面-太原站

架构是演化而来，而非设计而来

--没有完美的架构，只有不断演变、不断完善的架构。

一个架构设计的演进史

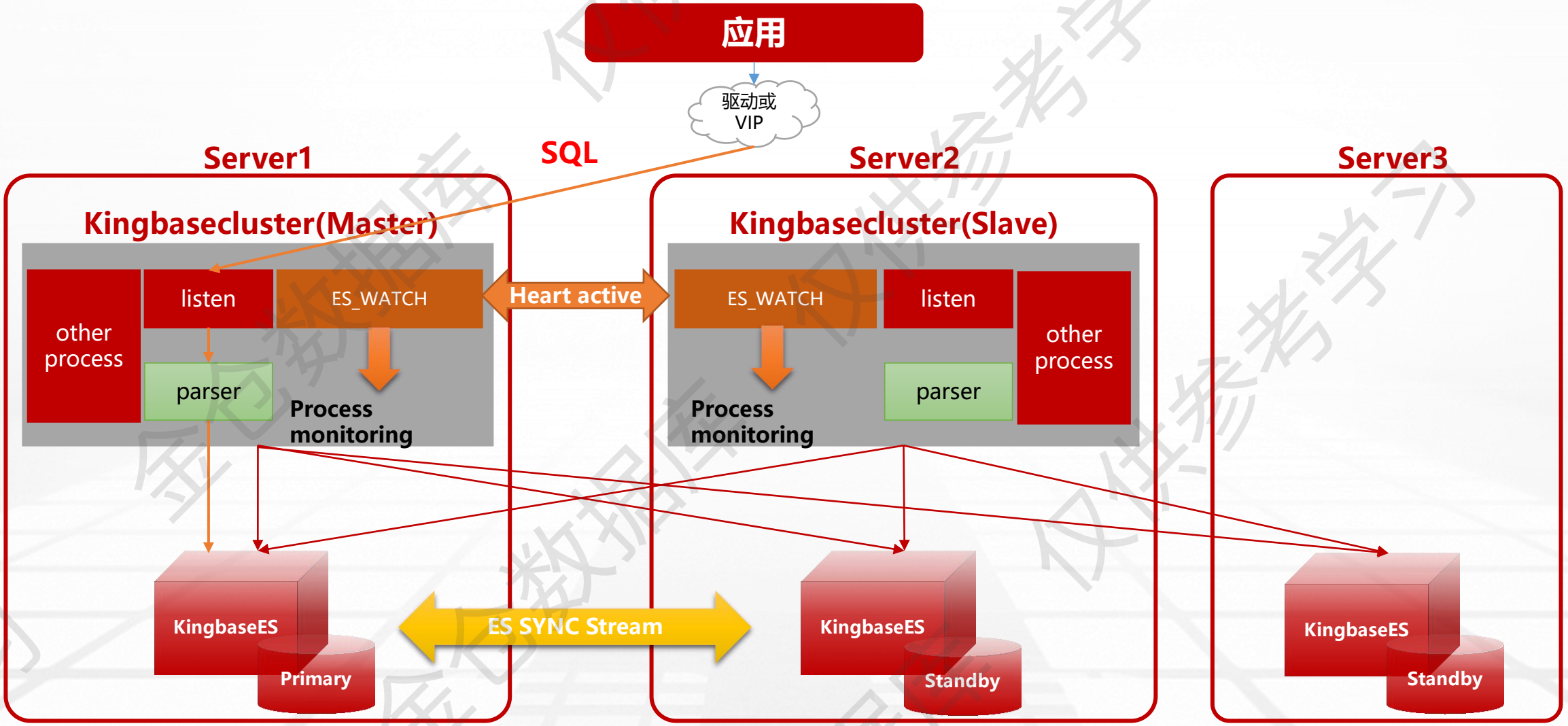
一个架构设计的演进史



→ 表示架构设计时的预估状态

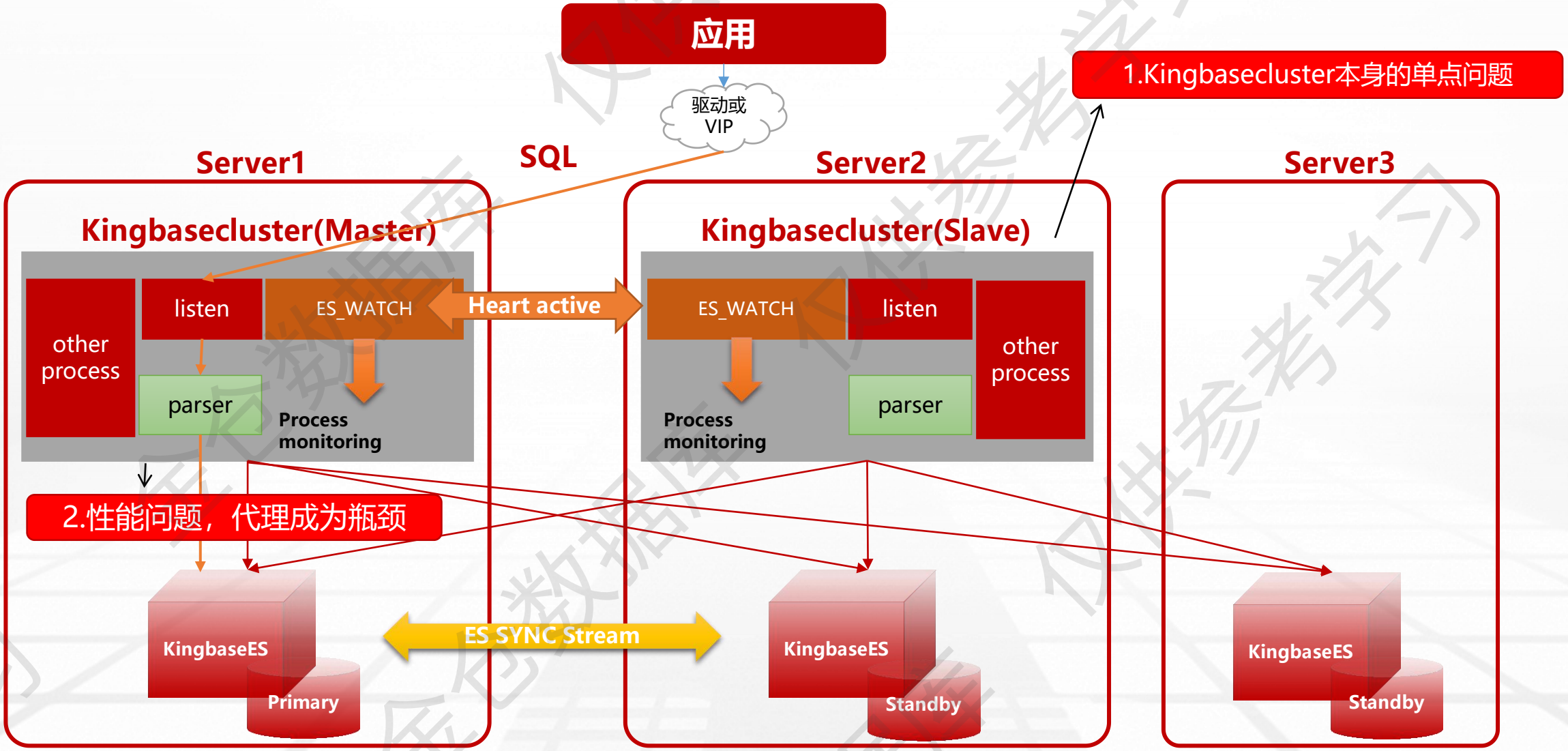
● 表示和架构设计无关，或因为需求、或因为用户行为数据、或因为BUG等均会对架构层面进行调整、改动

V8.2 (V1)RWC集群架构

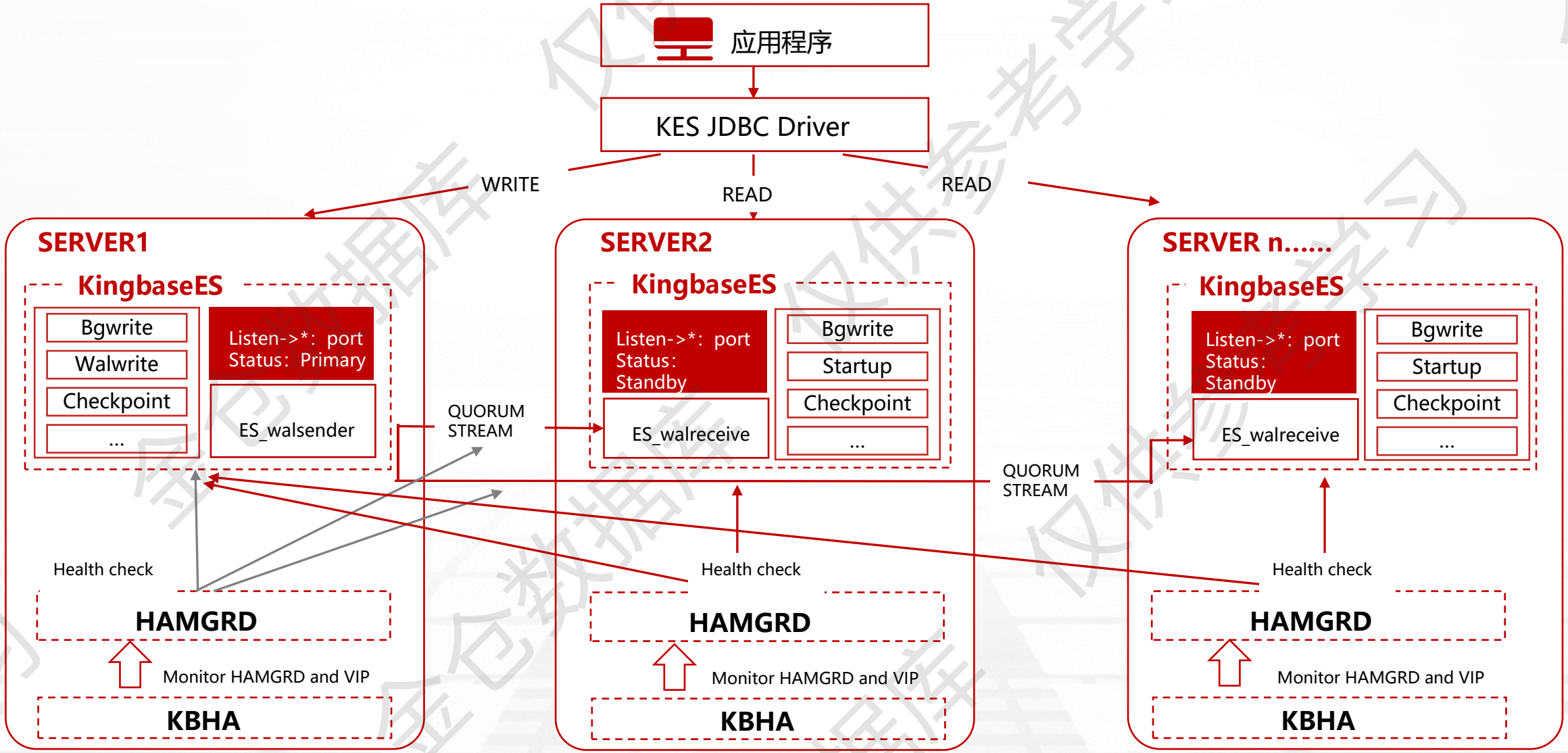


集群架构出现，解决了vip， 扩缩容，故障切换、恢复，读写分离。

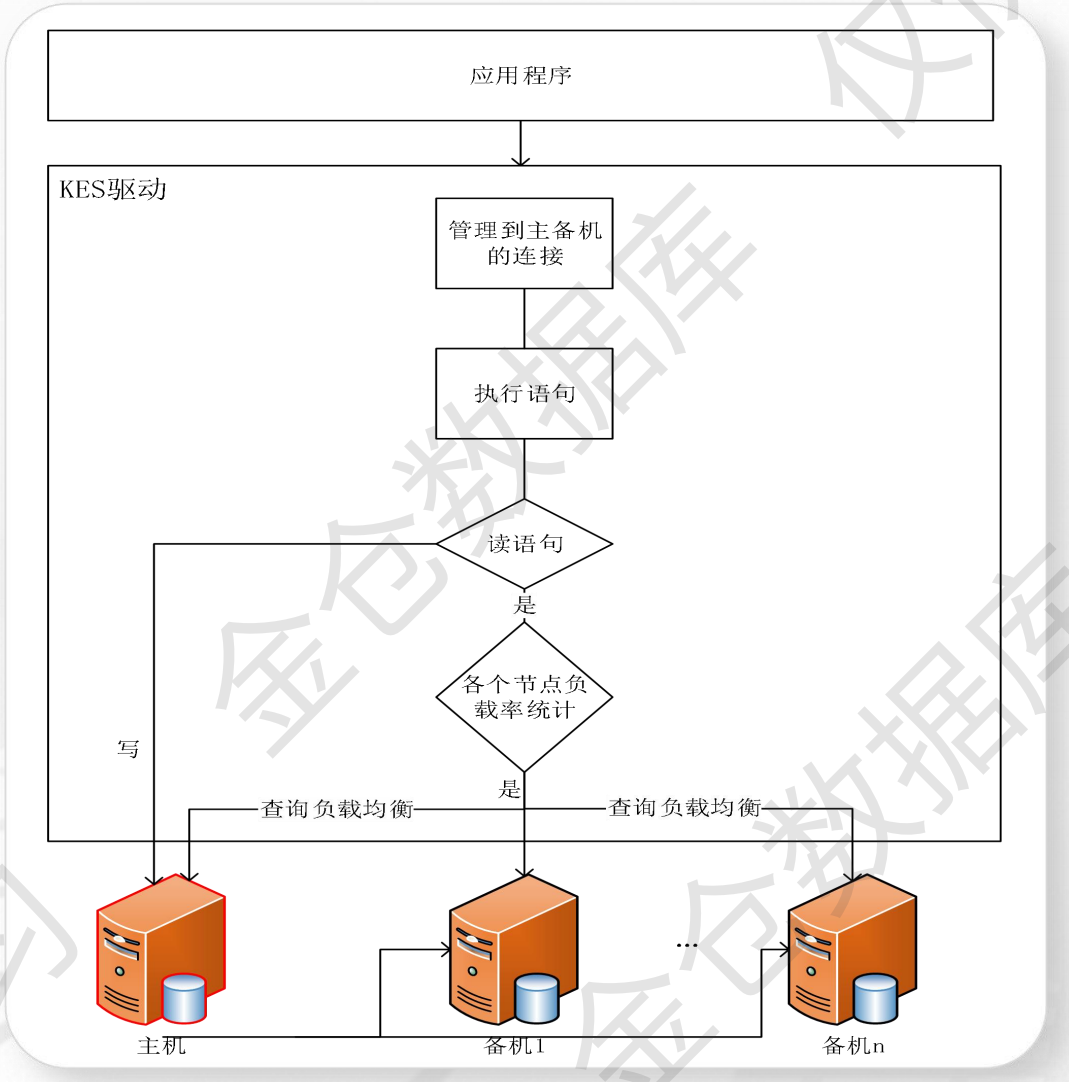
不太满足客户需求的点?


应用
驱动或
VIP
SQL
1.Kingbasecluster本身的单点问题
Server1
Server2
Server3
Kingbasecluster(Master)
Kingbasecluster(Slave)
other process
listen
ES_WATCH
parser
Process monitoring
Heart active
ES_WATCH
listen
other process
parser
Process monitoring
2.性能问题，代理成为瓶颈
KingbaseES
Primary
ES SYNC Stream
KingbaseES
Standby
KingbaseES
Standby

V8.6 (V2) RWC集群架构

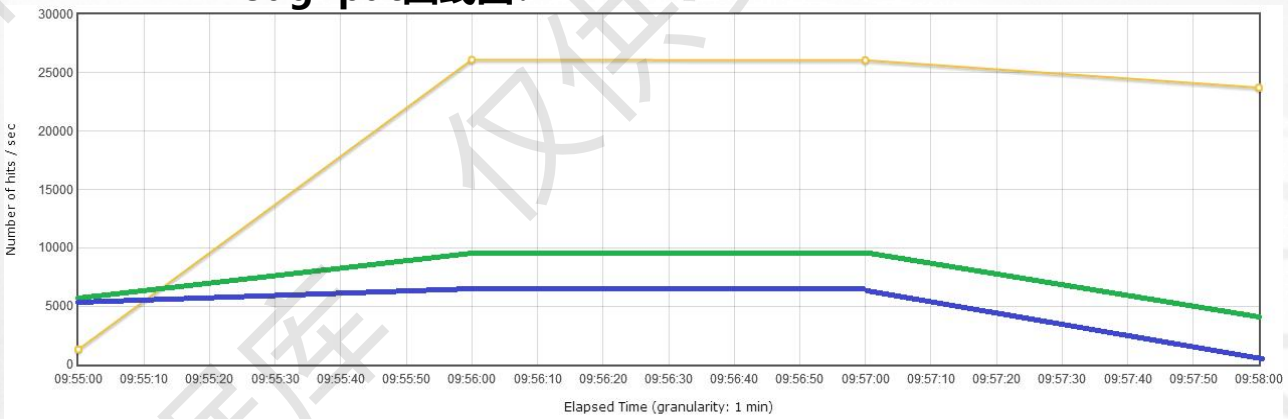


基于读写分离架构的读负载均衡

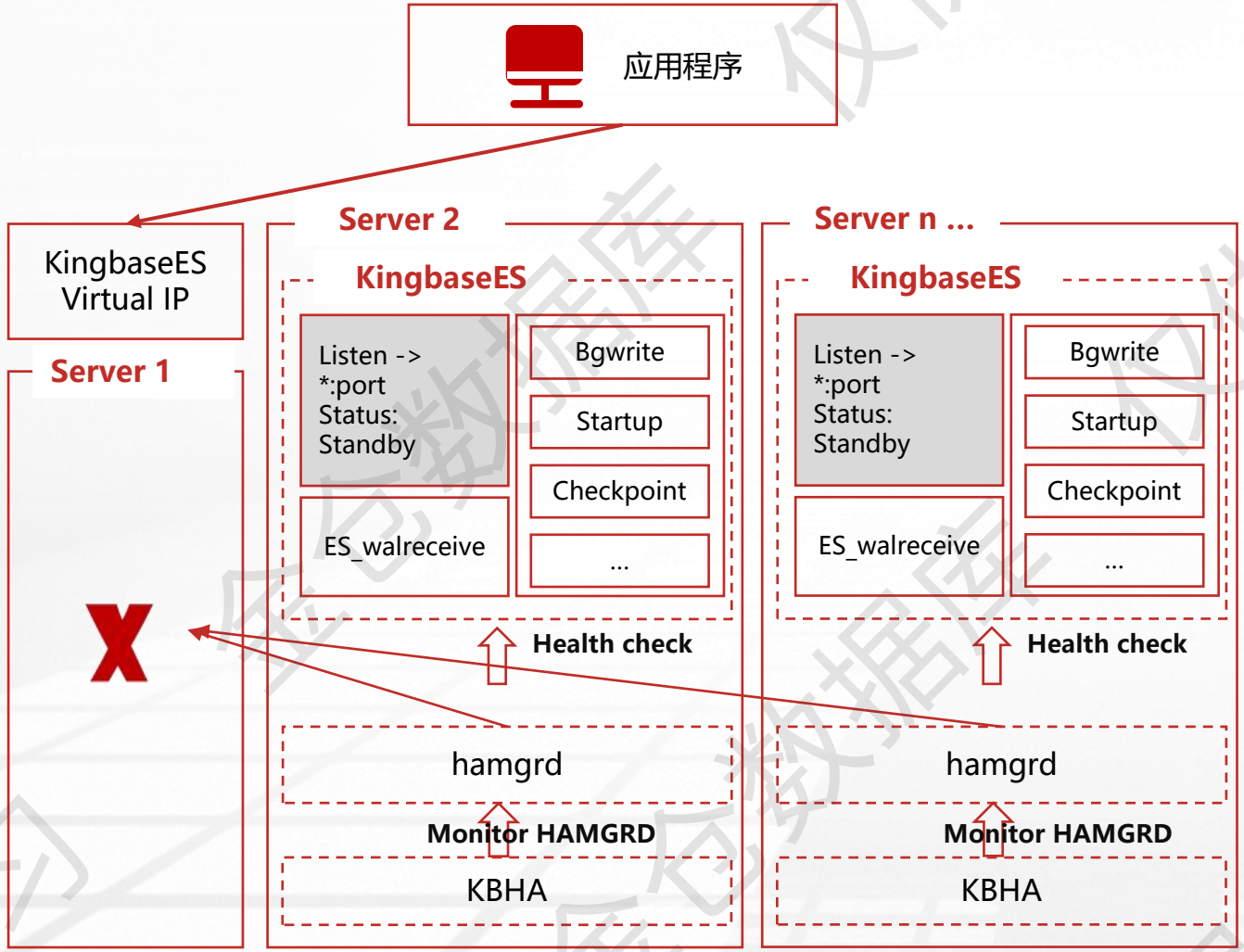


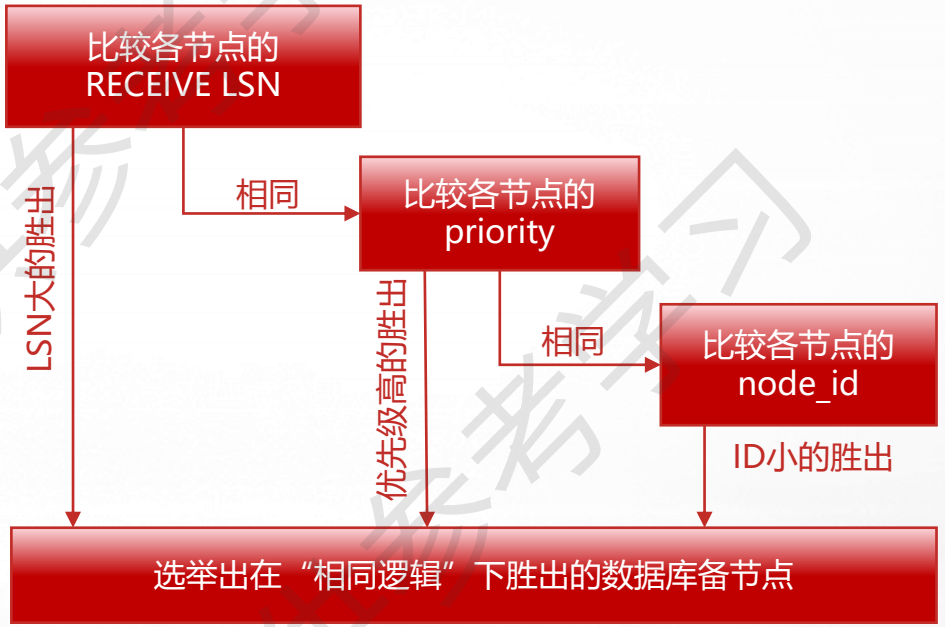
去除原有代理的性能瓶颈

- **读负载均衡：**
  - RWC所有的备机节点用于读操作，当我们读取操作较多时，驱动分发器将通过分发算法，将读操作均衡分布在所有正确的备机节点中执行，降低主节点负载，通过备节点负载均衡，提高查询性能。
  - **Jmeter 执行100万次select查询操作**
    - **JDBC读写分离对比单机性能提升1.5倍。**
    - **JDBC读写分离对比中间件转发性能提升3倍。**
  - **Throughput曲线图：**



解决管理组件的高可用问题：探测与选举 – 基于算法的 “最终一致性”





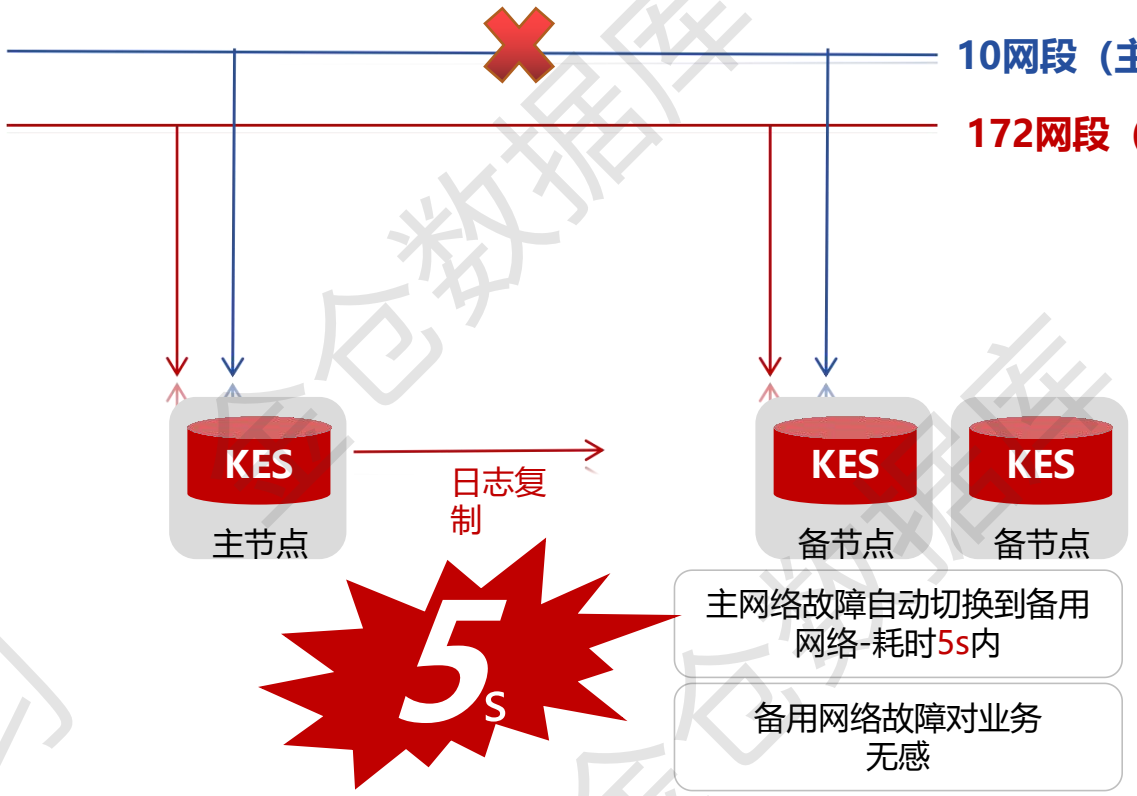
特点：

- 备节点的HAMGRd均使用 retry * delay 的时间间隔、 KCIping()方式检测主库是否故障，如果连接失败，则认为主库异常
- 在主库故障后，所有节点的选举过程做相同的处理逻辑，期间还会去其他备节点查询当前的lsn、优先级、节点ID
- 最后根据相同逻辑得出选举的“最终一致性”需被升级的节点
- 通过参数控制故障转移时间

客户需求、行为数据、运行效益等带来的可用性架构的能力增强

基于多链路冗余的快速切换技术

业务场景：轨交、新闻类等实时类业务，往往有着更低RTO（业务恢复时间）的指标，例如中断时间需控制在5s以内。



关键技术

- 多主机地址寻址
- 下链超时通知
- 链路动态缓存

技术适用范围

✓ 网络故障时，可以快速切换到备用网络继续提供服务

价值

✓ 集群之间利用多网络容灾
✓ 可以根据数据库此技术反向建设网络规划
✓ 减少因网络故障导致的服务中断

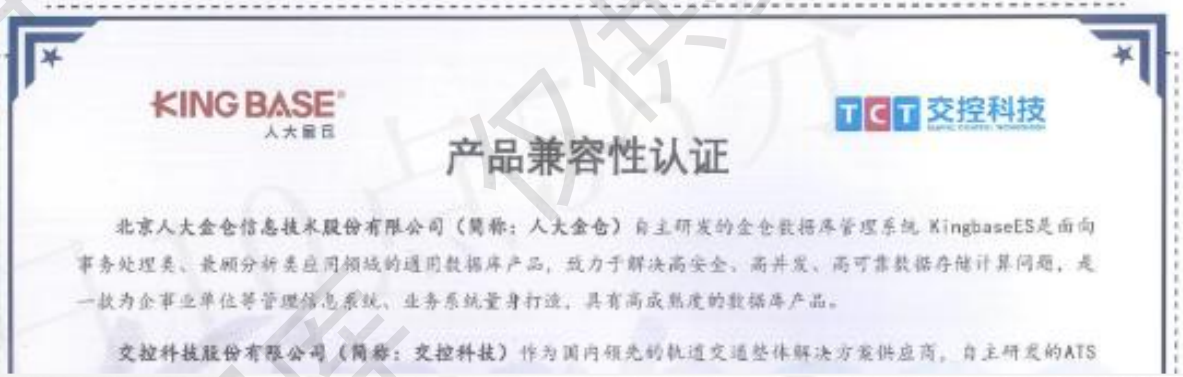
轨交客户案例

为符合我国城市轨道交通系统建设特点，满足高可用高性能高安全升级需求，金仓数据库首先实现了对Oracle传统双网方案的兼容，并且为保证双网方案具有较好的业务切换体验，还实现了IT环境解耦。**金仓自主研发的新双网方案**依托数据库主备集群的卓越架构实现了在双网段环境下的高可用性。数据库可直接接管双网卡，确保在任意一个网络发生异常时，网卡切换不依赖操作系统，无需协议转换，从而将性能损耗降到最低，**满足轨道交通5秒内完成双网切换的高可用需求。**

在网络与数据双重高可用性的保障下，贵阳地铁2号线能够**实现故障的自动检测与恢复，减少了人工干预的需要，极大提升了系统的自我修复能力。**这种智能化的故障处理机制，确保了地铁运行的连续性。

国产数据库首次工程应用

为响应“十四五”规划，推进自主化系统的研发与应用，交控科技针对ATS系统兼容适配国产化数据库进行了深入研究，对城市轨道交通7×24小时运行和故障应急处理等特性进行大量测试和软件架构优化，验证和提升数据库架构稳定性、故障切换及应急恢复便捷性，现已在深圳14、天津10、深圳12号线成功应用，同时在天津6号线二期、石家庄2号线运营线路完成无缝切换升级。信号系统首次应用国产化数据库，为轨道交通行业数据库增加了更多选择。


KINGBASE 人大金仓

TCT 交控科技

产品兼容性认证

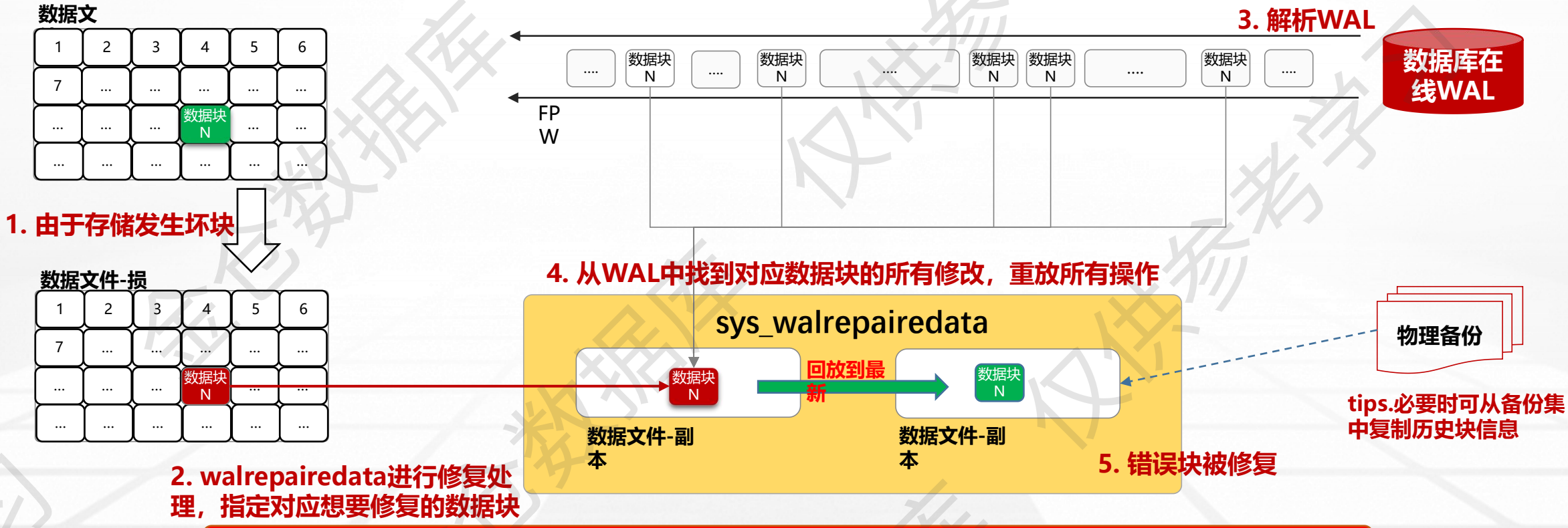
北京人大金仓信息技术股份有限公司（简称：人大金仓）自主研发的金仓数据库管理系统 KingbaseES是面向事务处理类、数据分析类应用领域的通用数据库产品，致力于解决高安全、高并发、高可靠数据存储计算问题，是一款为企事业单位等管理信息系统、业务系统量身打造，具有高成熟度的数据库产品。

交控科技股份有限公司（简称：交控科技）作为国内领先的轨道交通整体解决方案供应商，自主研发的ATS

基于WAL日志重放精准修复文件或数据块

KINGBASE | 金仓社区

业务场景：当用户存储异常损坏了部分数据块或某个表文件时，为了修复损坏的数据，如果基于物理备份恢复全部数据则耗时久，业务中断时间长。提供基于WAL日志重放的方式，能够精准的修复数据块或文件，快速完成数据huif.

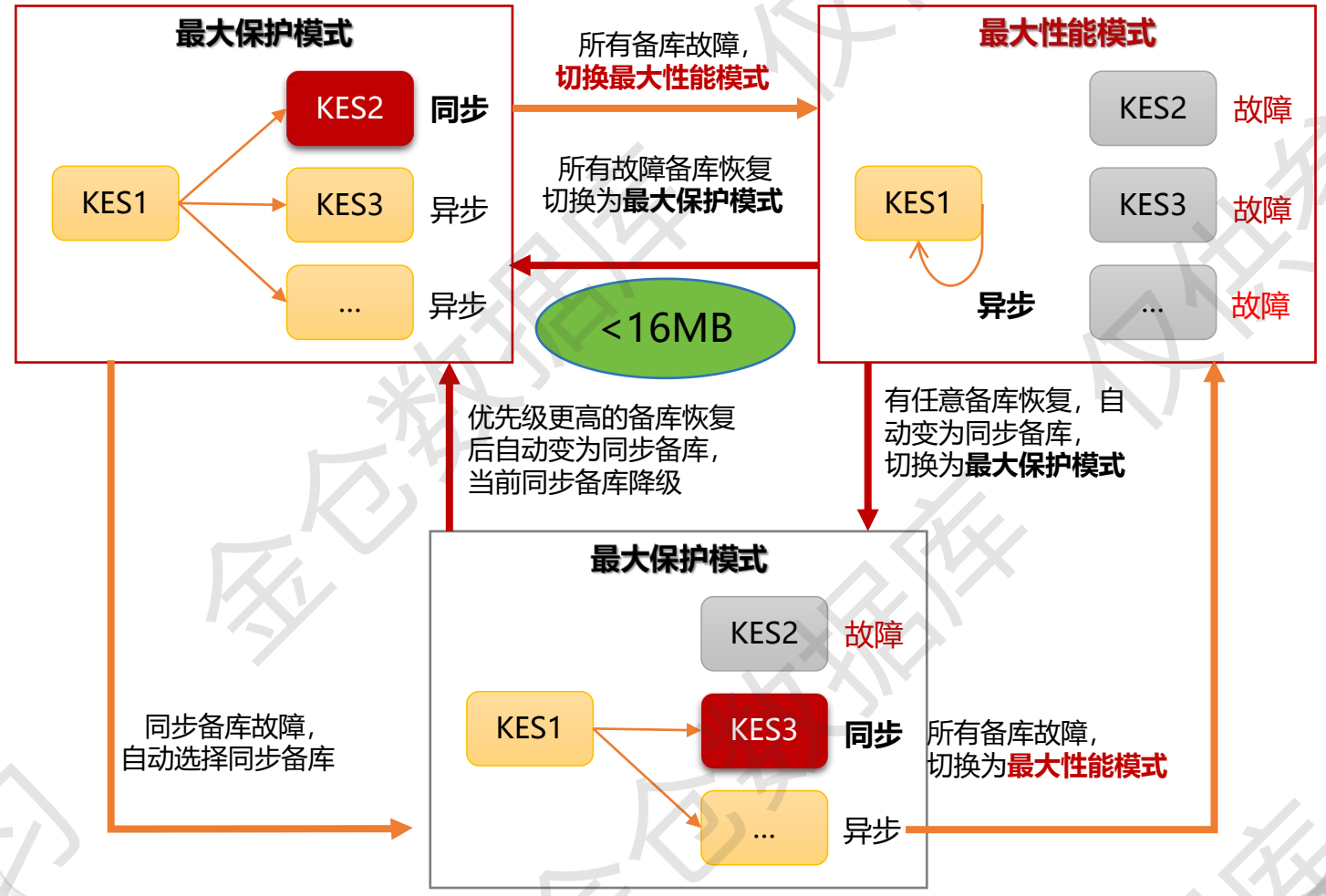


解决客户在磁盘块发生故障时，一种低成本高效的恢复手段

技术适用范围：

只要保存有修改损坏数据块的WAL日志，或者有物理备份存在，即可快速修复小范围的数据损坏——多个数据块损坏、或某个数据文件损坏。

集群同异步动态调整



业务场景：

- 所有业务场景中，用户设置同步模式时，同步备库异常下线导致的业务hang住
- 异步变同步的人工参与过程

价值：

- 同异步动态切换，在条件满足时任然保持RPO=0：

➢同步变异步：自动，**保护应用不会hang住**

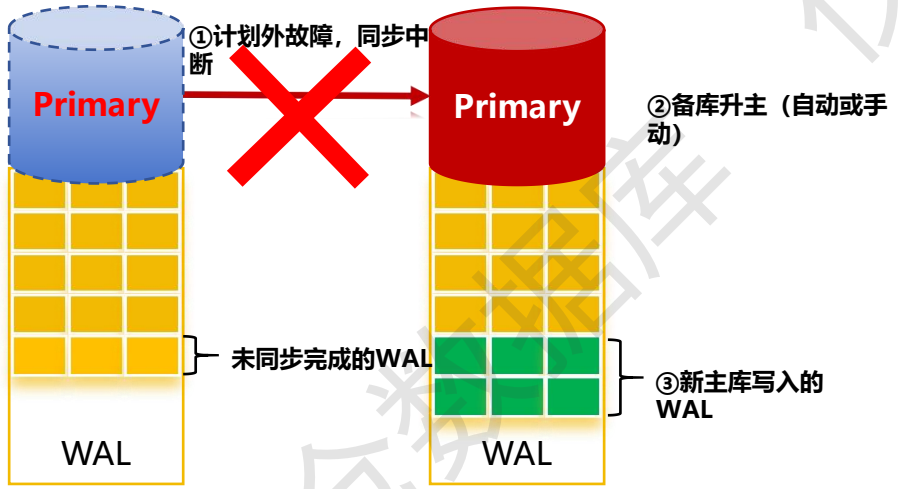
➢异步变同步：自动，**保护应用不会hang住**

解决在集群架构的各种状态转换时对客户应用造成的影响

基于差异点WAL日志回滚的主备集群恢复技术

KINGBASE | 金仓社区

计划外故障：有差异WAL日志



业务场景：在集群主备切换数据出现可能的分叉时，需要重新组成集群

关键技术：

- **基于WAL快速定位**

  读取本地和主库的WAL日志，对比检查点快速找到分歧点，并根据分歧点找到差异数据。

- **回滚差异数据**

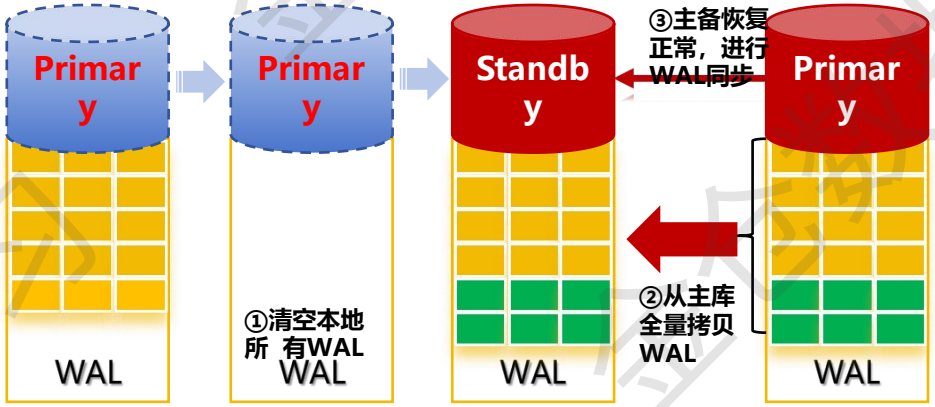
  回滚差异数据，增量同步主库的WAL日志和数据到本地，启动数据库后即可作为备库与主库进行数据复制。

技术适用范围：主备集群中，数据库故障且有差异数据（WAL日志）时。

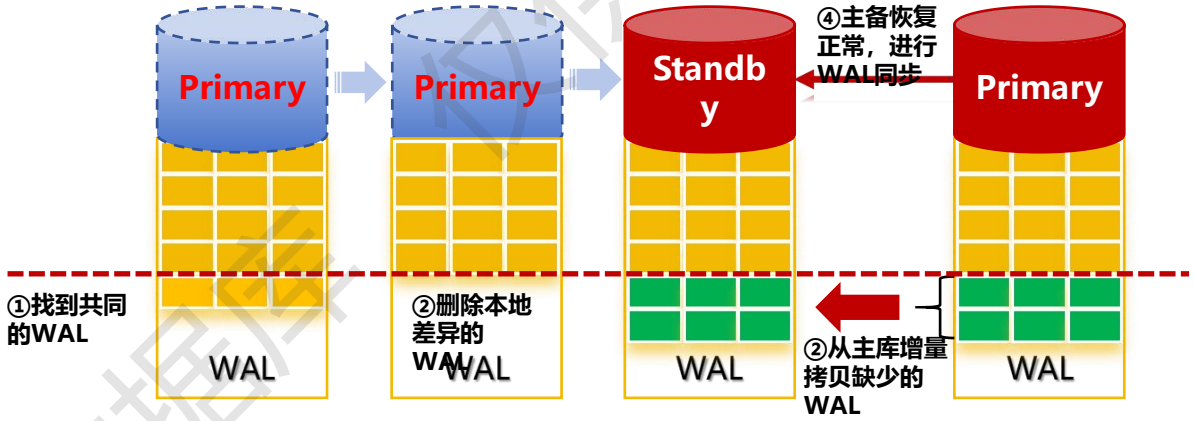
价值：能够显著提升故障数据库的恢复速度（**恢复时间减少50%**），减少WAL文件拷贝（**WAL拷贝减少 60%**）

优化前	优化后
传输数据量：5899 MB	传输数据量:363 MB
时间：277.809187 秒	时间：22.536810 秒
传输Wal文件数：343	传输Wal文件数:13

优化前修复故障数据库：全量拷贝WAL



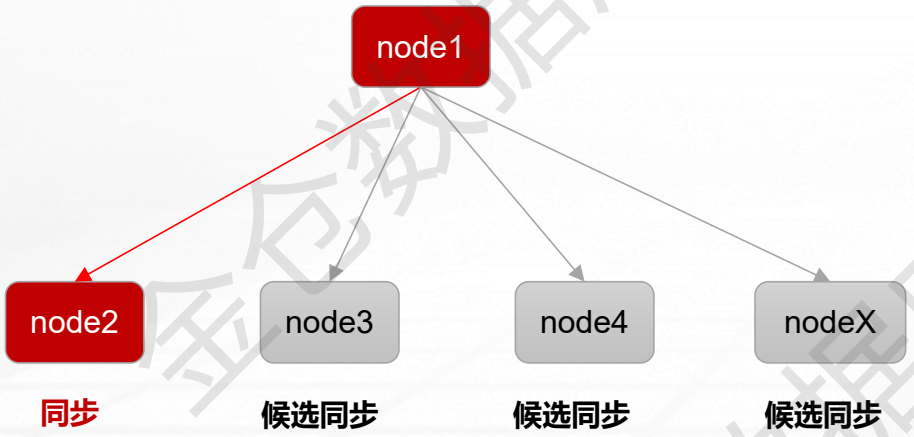
优化后修复故障数据库：差异回滚+增量同步



基于用户动态规划的主备集群同异步技术

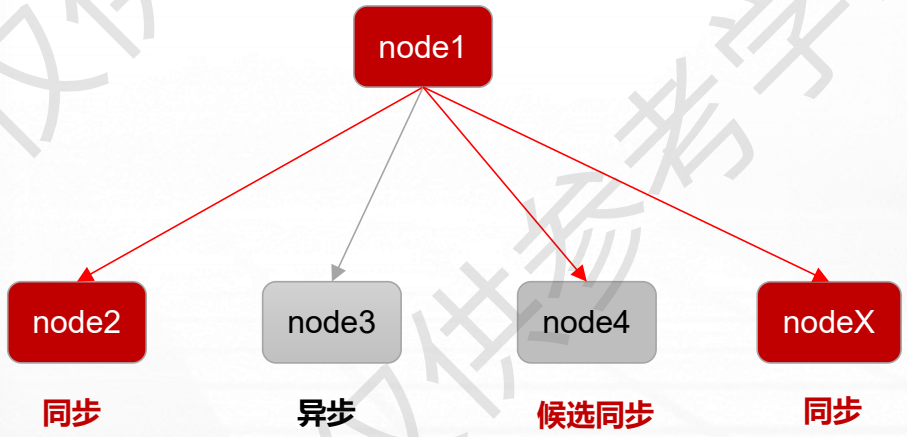
业务场景：在数据库集群中，客户在规划整个容灾系统时，可能要求某些节点应该作为同步节点保证RTO为0、某些节点应该为异步节点保证不影响主库性能的前提下进行数据复制，或者需要多个同步节点保证更高的可用性。在此场景下，主备集群提供更高自由度的配置，支持用户自定义每个节点为同步模式或异步模式。

Before



只有第一个节点固定为同步，其他节点角色固定

After

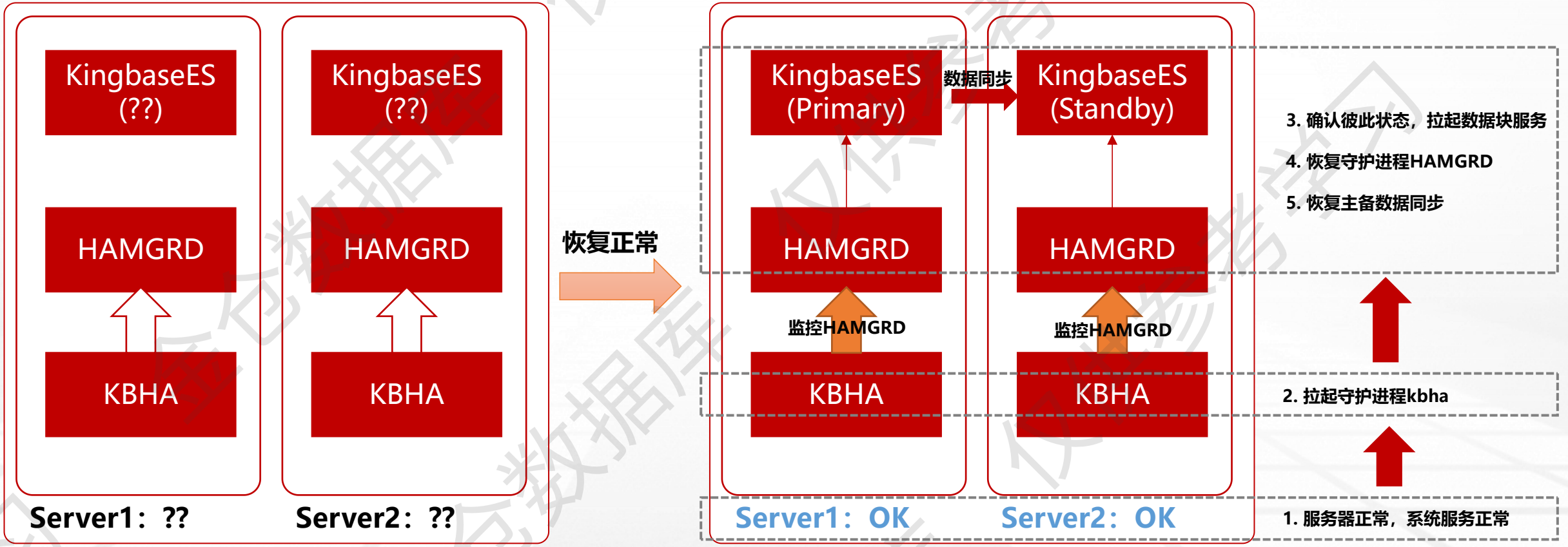


可以指定每个节点的角色状态

解决用户有特定的同异步需求时，例如个别服务器性能高-要求多节点同步；部分服务器性能差，只需求异步，状态固定

集群全故障自启动技术

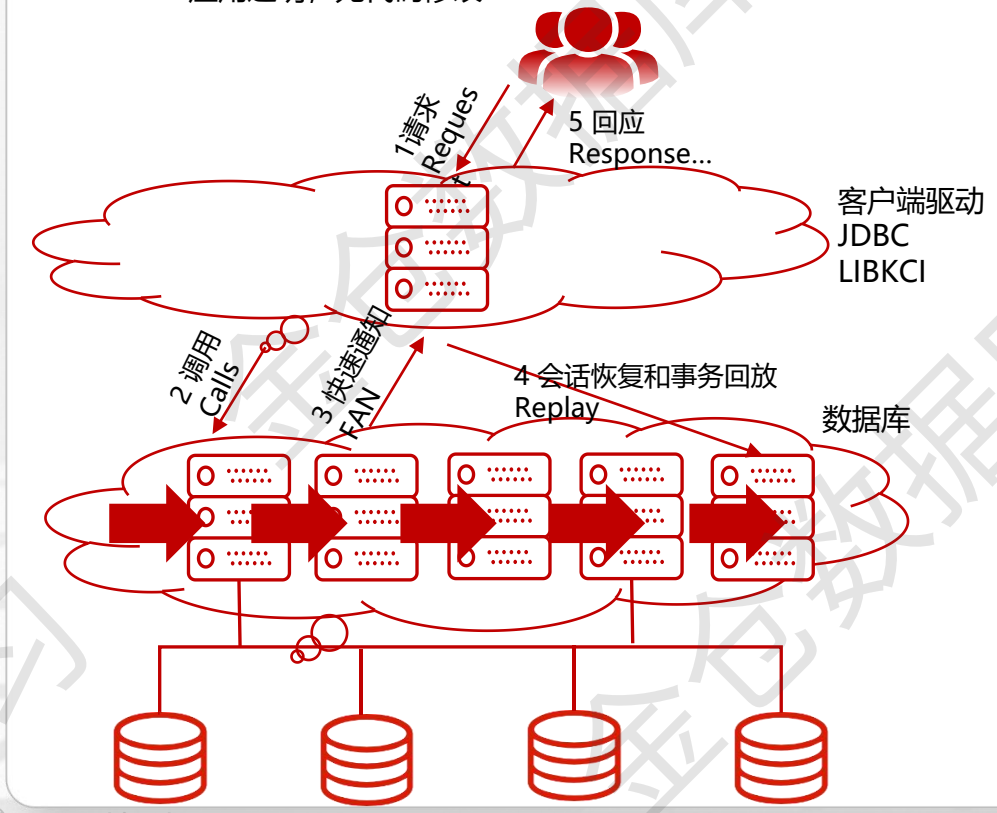
业务场景： 出现断电、关机等所有设备都停止运行的场景下，一旦服务器恢复正常，集群守护进程可以自行启动、自动探测数据库状态并恢复正常。



技术适用范围： 出现全局故障的场景下，服务器恢复正常，集群即可在不需要人为介入的情况下快速恢复正常。

基于透明应用连续性 TAC的应用无中断

- **数据库变更 故障？客户无感知，畅享稳定体验（TAC）**
  着重用户体验
  - 屏蔽计划内维护和意外停机
  - 应用透明切换到冗余节点
  - 自动恢复进行中的事务，屏蔽数据库层故障
  - 数据库中断对用户来说只是延迟执行
  - 有效掩盖滚动打补丁，网路故障，实例崩溃，主备切换等中断时间
  - 应用透明，无代码修改



从HA到TAC的演进

高可用性 (High Availability)	透明应用连续性 (Transparent Application Continuity)
最小化停机时间	无需停机时间
透明故障转移TAF	运行中的事务可获得保护
停机会导致运行中的事务中断	Rolling维护工作对应用是透明的
数据库可以Rolling维护	屏蔽绝大多数错误
错误可能“可见”	在HA的基础上实现
为单点故障设计	

TAC的阶段：正常阶段和重放阶段

正常操作阶段	中断重放阶段
客户端标记请求	检查重放已启用
客户端捕获原始调用、它们的输入和验证数据	创建新连接
服务端决定哪些可以重放，哪些不能重放，禁用副作用	验证及时性
在请求结束时，客户端清除队列，为下一个请求做准备	服务器端检查重放是否有效
	服务器端检查事务是否提交，如果未提交则回滚
	客户端重放捕获的调用
	服务器端和客户端确保返回到应用程序的结果与原始结果匹配
	成功重放后，客户端将控制权返回给应用程序

基于低热点拷贝+Feedback节点元信息的无感数据库弹性伸缩

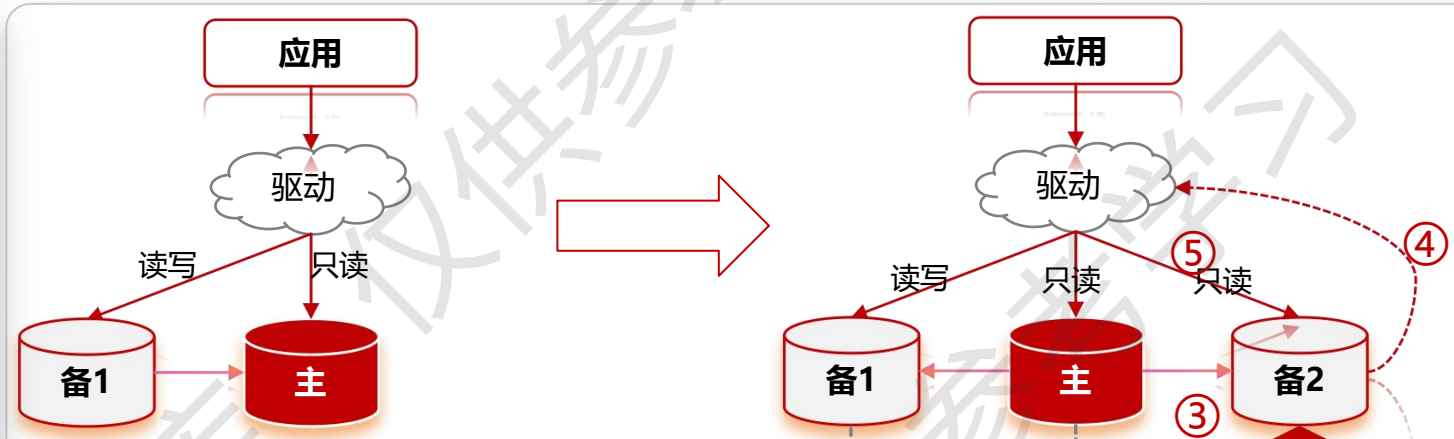
业务场景：随着业务系统的不断运行，业务负载通常也会产生一定的变化，同时需要数据库集群节点也相应的进行动态调整：如果负载增多，则需要增加数据库节点；如果负载减小，为了节约成本而缩减数据库节点。KingbaseES RWC支持数据库节点在线无感的进行弹性伸缩，此过程中不影响在线业务。

技术方案

- 从备库或物理备份低热点节点拷贝数据，最大程度减轻对主库的影响
- 异步平稳追平增量数据，对主库降低影响
- 集群节点变化反馈给驱动，驱动识别节点变化并重新进行负载均衡

效果

- 扩容（或缩容）过程中，集群负载稳定无影响，业务层无感知
- 扩容（或缩容）后，驱动识别到节点数量变化，自动重做负载均衡，业务层无感知



扩容：

① 从低热点备库或物理备份中拷贝基础数据

② 从主库Catchup落后的新增数据

③ 追平数据后，可选择以同步或异步模式加入集群

④ 数据层Feedback元信息至驱动层

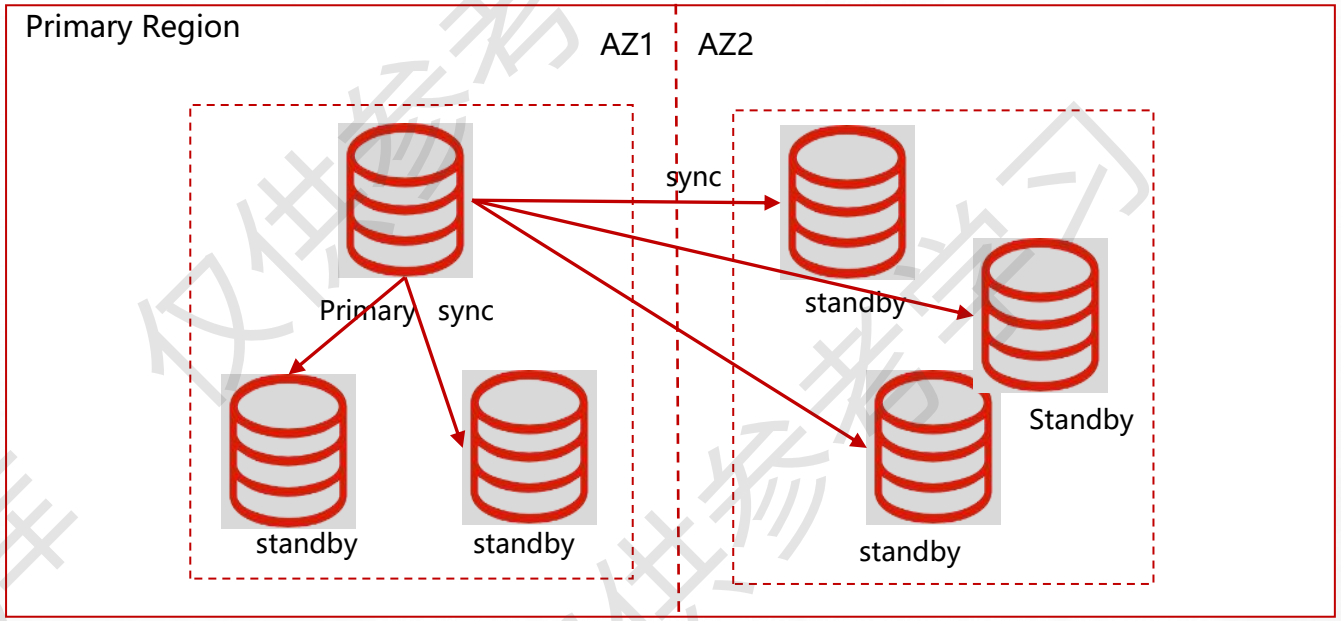
⑤ 驱动根据节点信息和负载信息重做负载均衡

不太满足客户需求的点?

传统架构在异地容灾的问题

- 主库压力负载过高：**高IO**
- 中心间带宽压力高：**高带宽***N条线
- 备份机制不全，备份只能在其中一端实现
- 可用性机制不全：依赖本地的“仲裁节点”，无法处理分区，**极易发生双主**，或者无法切换的场景

行业深水区：DR远距离容灾时



缺少CAP的P

高可用容灾-架构技术改进思路

传统架构在异地容灾的问题

- 主库压力负载过高：**高IO**
- 中心间带宽压力高：**高带宽***N条线
- 备份机制不全，备份只能在其中一端实现
- 可用性机制不全：依赖本地的“仲裁节点”，无法处理分区，**极易发生双主**，或者无法切换的场景

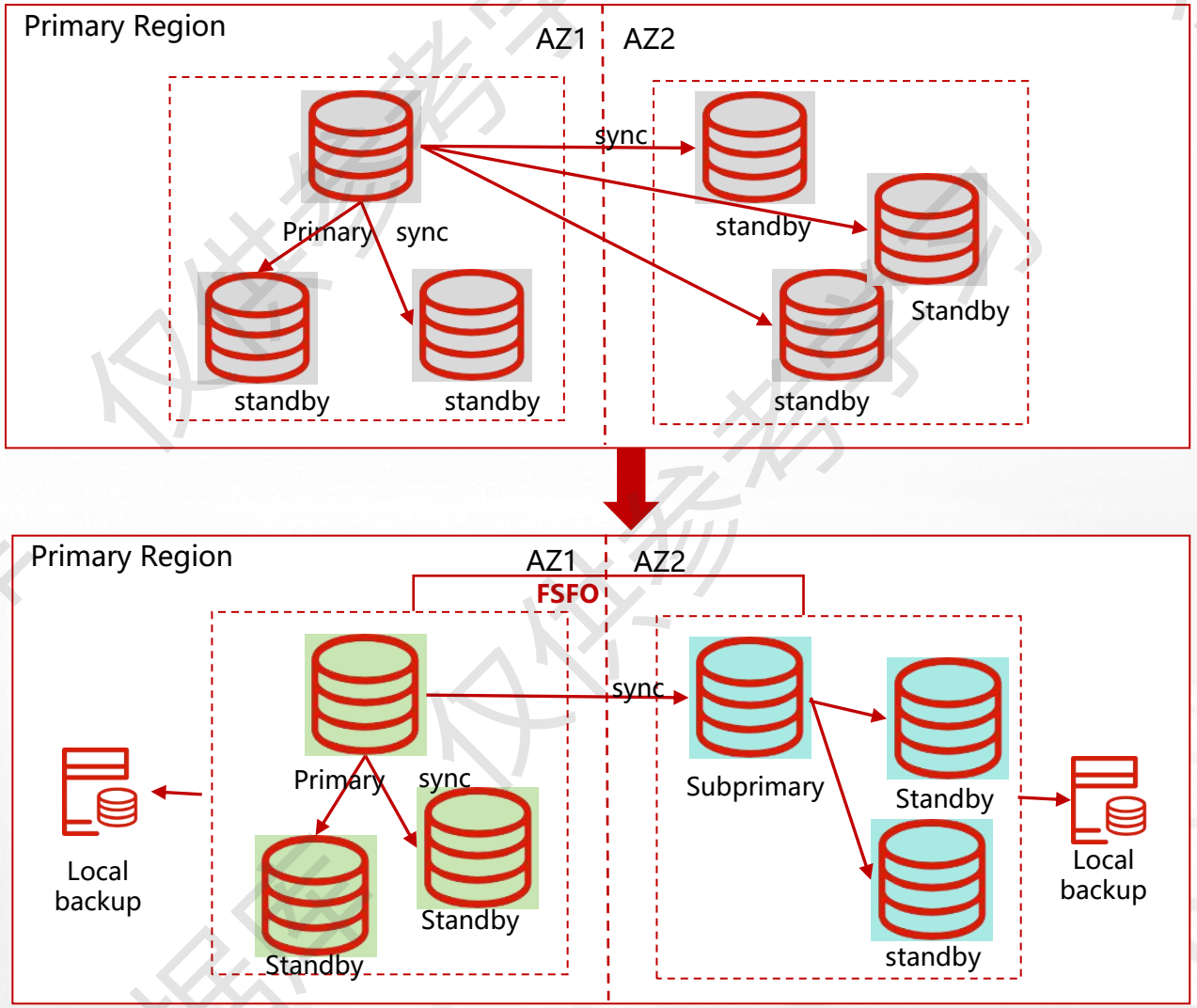
理论依据：分治法+引入CAP组件

① 大集群拆成若干小集群，引入**Subprimary**角色，级联复制，降低中心间的带宽压力和主库压力

② 每个集群，退化到本地高可用，可以有**独立的物理备份节点**，且集群内部只管理**中心内的故障切换**

③ 中心间的FSFO故障切换在更上层模块处理，不在集群内部

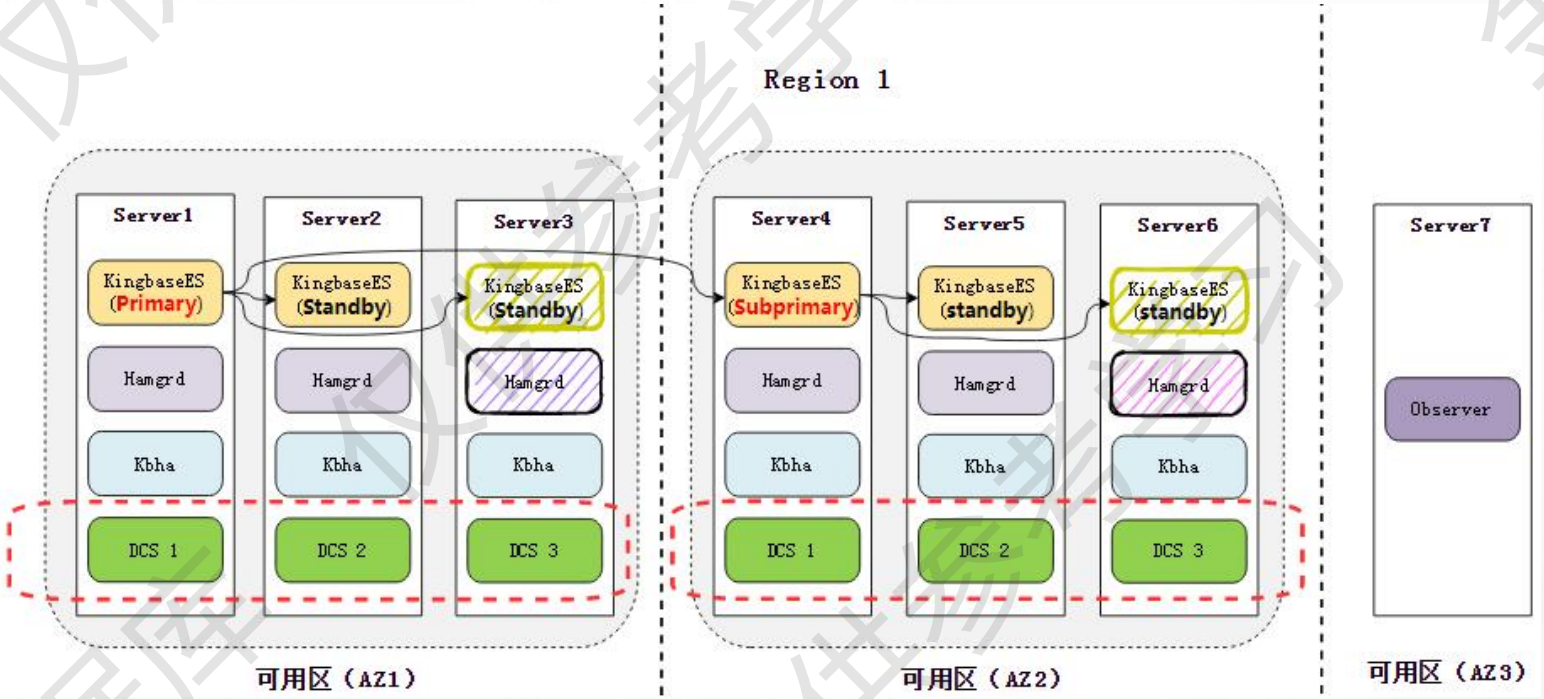
④ 集群架构引入CAP，更好的处理分区



V9 (V3.0) 架构变化

引入分布式组件DCS

- **DCS**：Raft协议，存储集群节点元信息
- **选主机制**：数据库的主只在多数派节点中产生
- **Fence机制**：少数派节点会自停服
- **自动恢复**：节点上电后自动恢复服务



故障中心间仲裁组件Observer

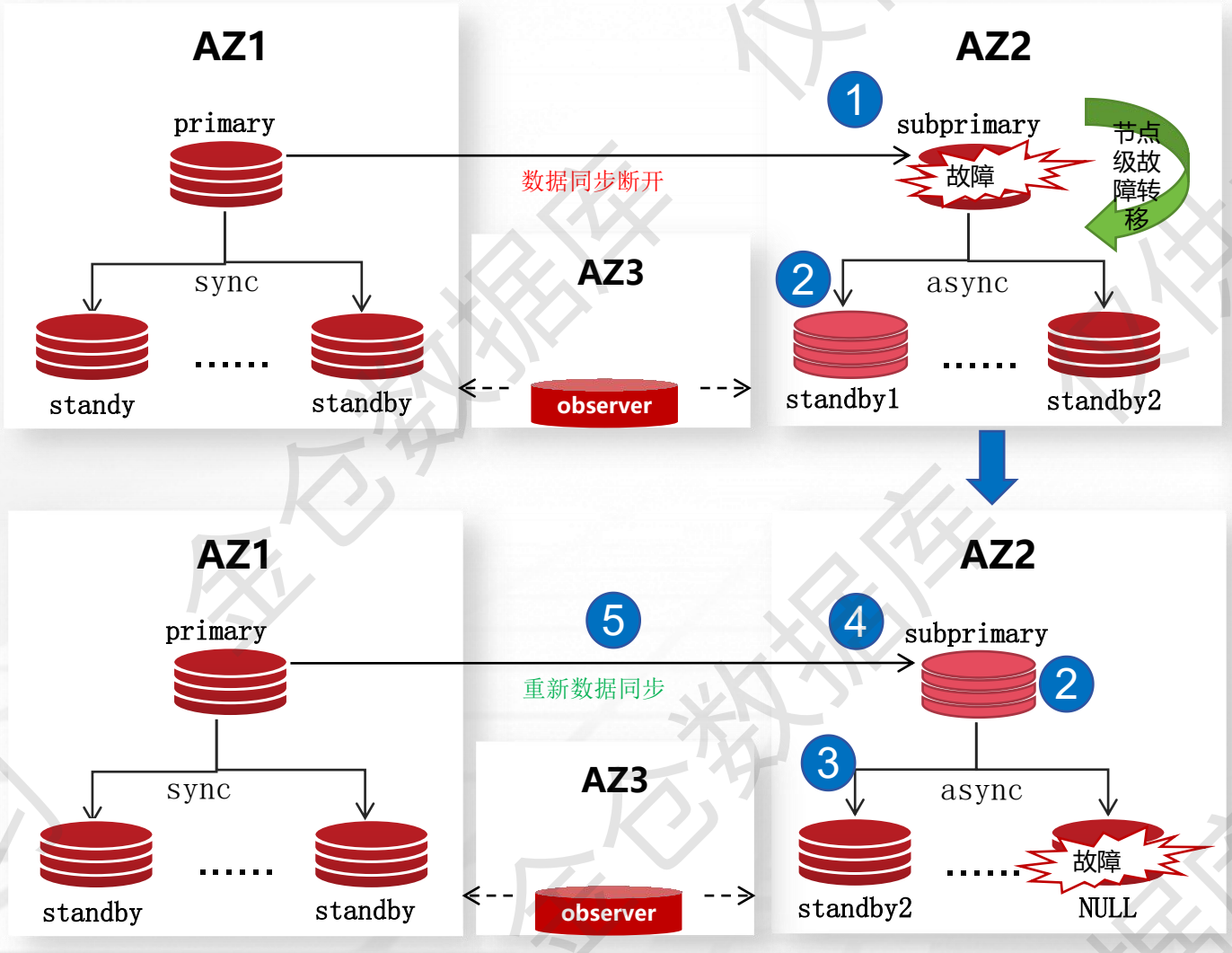
- AZ1故障时，则会在A2上选出新的主进行服务

高可用指标（默认）

中心内：RTO=秒级（30s）、RPO=0

跨中心：RTO=分钟级（60s），RPO=0 （极限情况不为0）

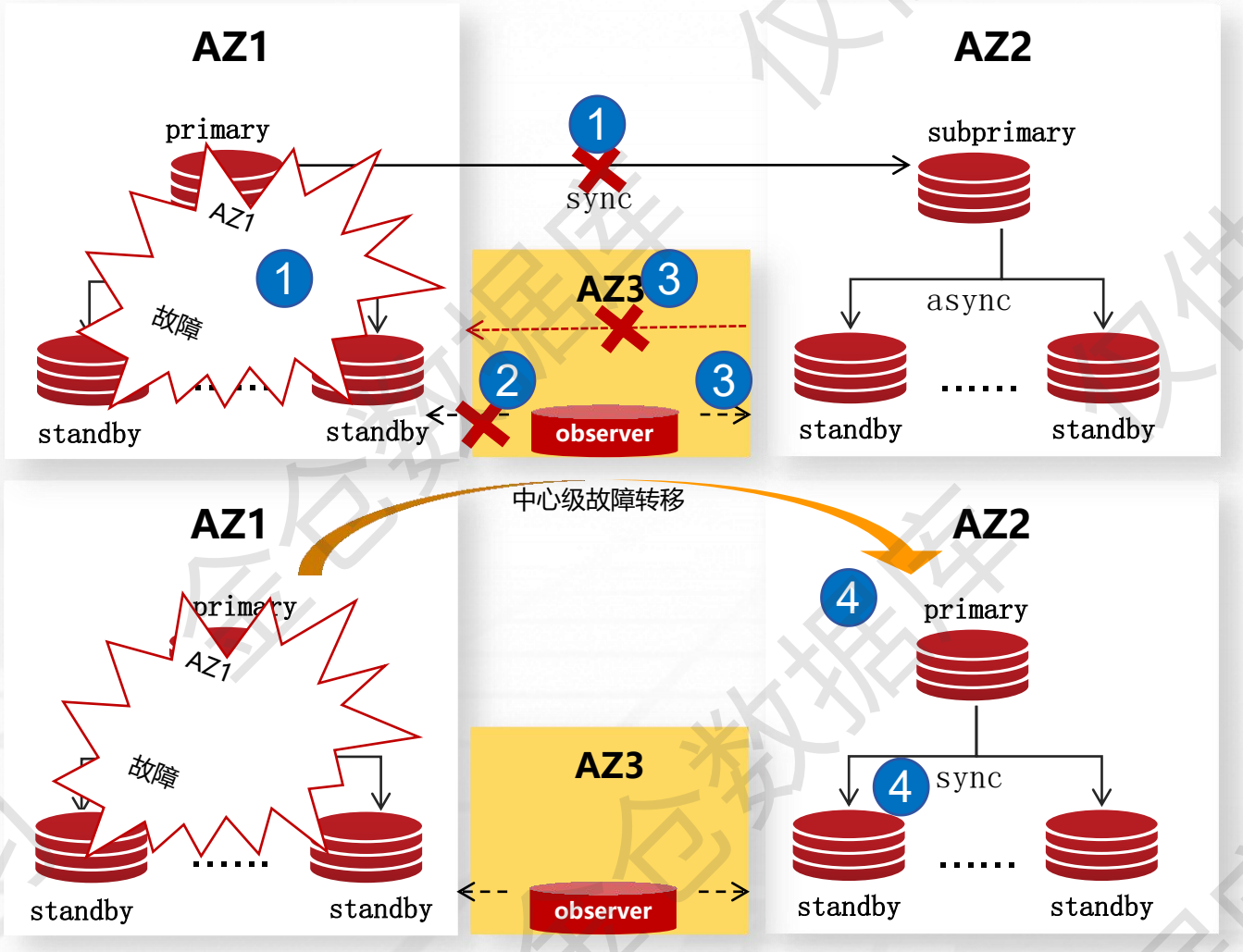
中心内故障 示例



中心内节点故障场景

1. AZ2的Subprimary节点发生故障
2. 其中一个级联节点**提升为新Subprimary**节点
3. 新提升的Subprimary节点会将所在区域内的其他节点变为自己的standby
4. Subprimary节点会定期检测与primary的链接关系，断连后重新链接；
5. 恢复数据同步

中心级故障 示例



中心级故障转移场景

1. AZ1发生故障，中心间同步断开
2. observer诊断到与AZ1连接断开；
3. **Observer通过AZ2的节点访问AZ1失败；(*重要)**
4. Observer将AZ2的subprimary节点提升为primary主节点，AZ2的级联节点自动跟随新主节点成为新的standby，并由**异步切换成同步模式。**

案例1 - 某期货交易所

场景介绍

基本情况

金融期货交易所核心交易系统替代

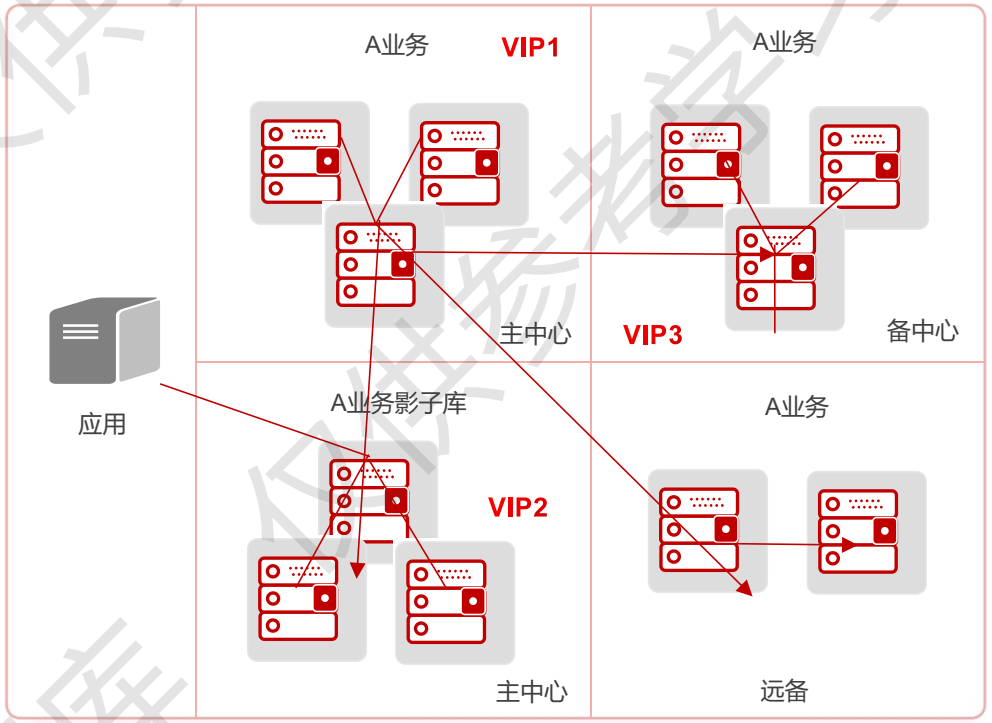
规划3中心容灾方案（同城30KM，远备 900KM ）

关键问题

- 业务可靠性要求高，需要**3地3中心容灾**
- 业务读负载高，且有低延迟读需求，需要横向读负载扩展
- **分层管理，性能良好，满足大量并行查询，休市跑批**

解决方案

- RWC多集群容灾方案解决单集群主机流量过大问题，支撑**11节点容灾集群**，多中心业务就近读**实现低延迟**



案例2 - 某省纪律检查委员会

场景介绍

基本情况

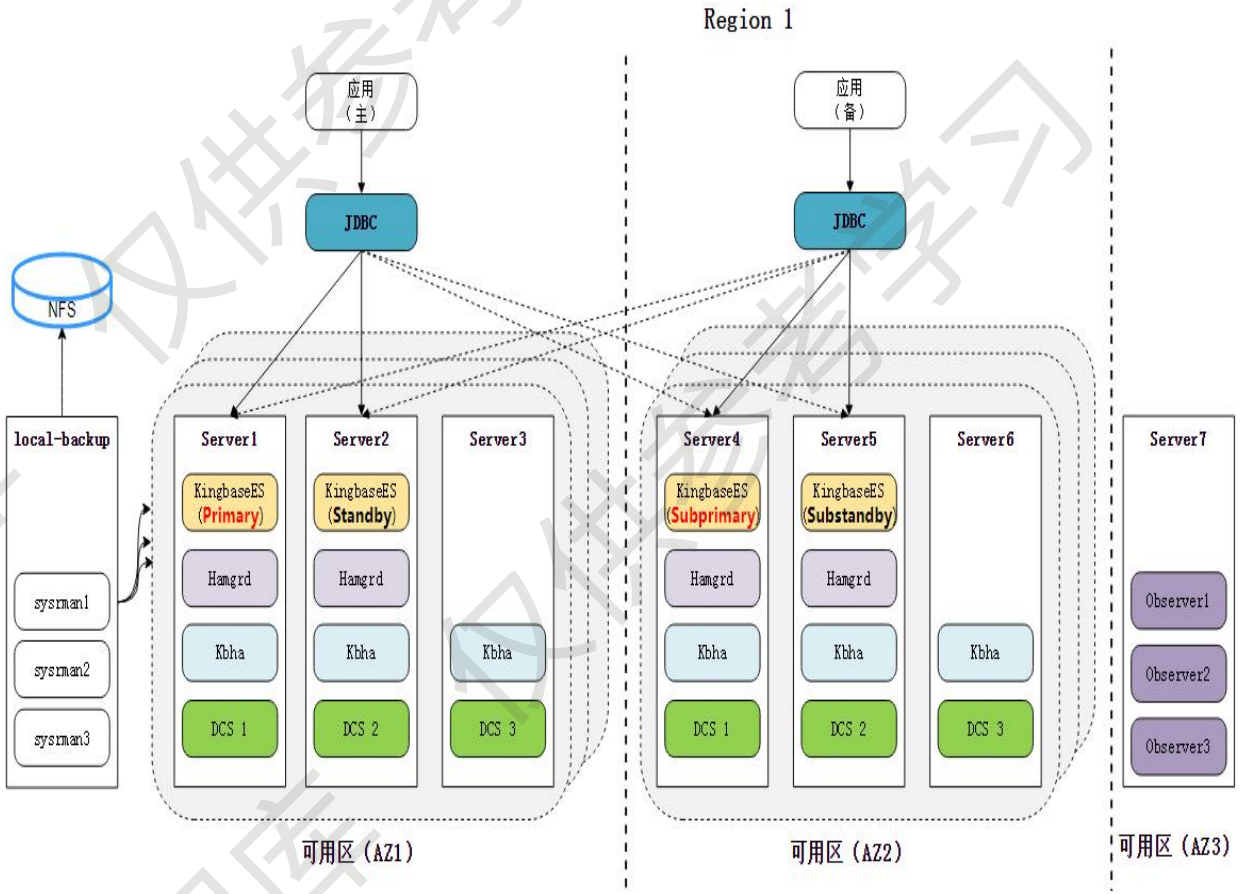
50套核心业务系统国产化替代

规划3中心容灾方案（同城81KM）

关键问题

- 业务可靠性要求高，需要**50套业务系统分散在不同集群中**
- **中心间不能丢数，且需要自动切换的能力**
- **有集中备份要求**

解决方案



总结对比

多中心容灾架构	金仓	未做建构改造的集中式厂商A
主库负载压力	**低（1拖N/2）**	高（一拖N）
中心间带宽	**低（带宽仅一条链路）**	高（带宽=灾备中心数量M）
中心间切换脑裂风险	**低（增加中心级别观察器角色）**	高（无独立中心级别观察器）
中心内可用性能力	**高（全故障自动恢复）**	低（人工运维介入多）
性能	**优 > 30%**	低

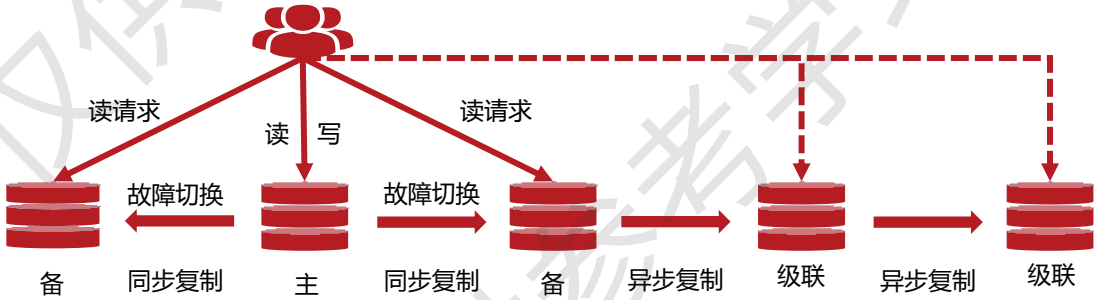
高可用 - 从实例、集群到多中心的高可用保障

实例级

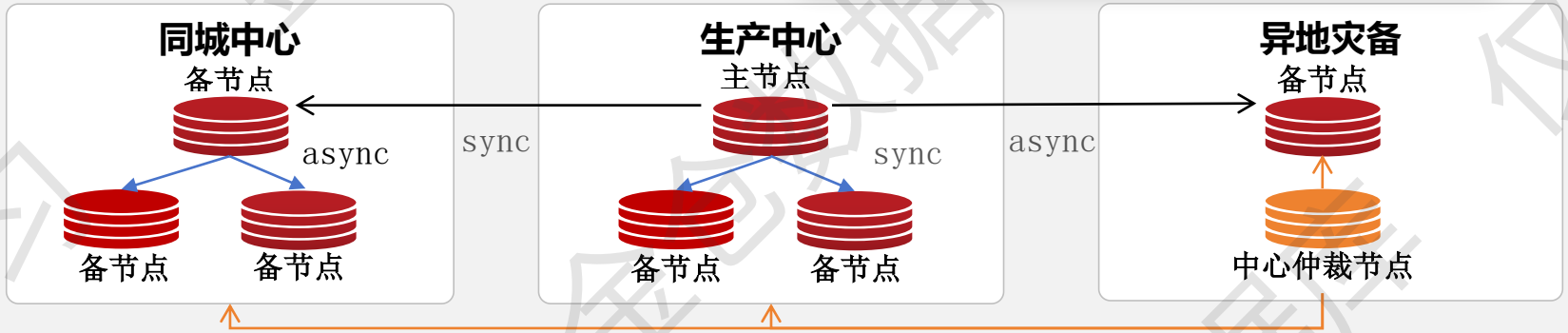
全量/增量/差异备份	全库/用户/对象备份	文本/二进制/加密输出	远程备份	备份三方平台集成
全库/事务/时刻恢复	全库/用户/对象还原	文件系统防误删	闪回数据库	闪回表
控制文件多副本	数据一致性校验	wal解析成SQL/undosql	wal在线修复data	进程自故障自恢复

集群级

- 基于物理日志的全同步复制，数据不丢失
- 同异步自动转换，不阻塞事务，业务持续可用
- 备机日志回放并行加速，故障快速切换
- 增强型一致性协议选主，避免脑裂
- 节点故障恢复后自动回归，保持集群规模稳定



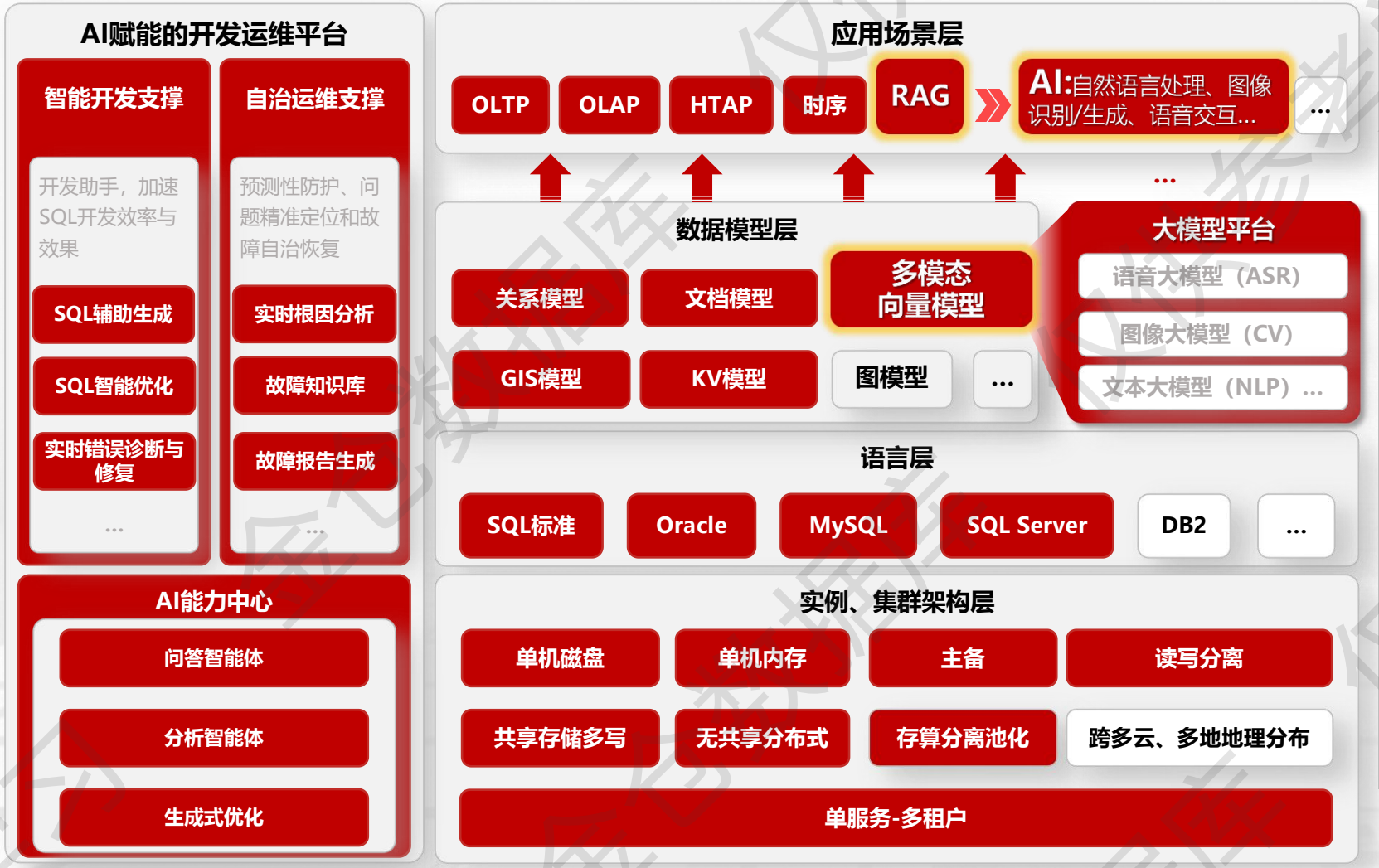
中心级



- 同城双中心物理全同步，确保数据“零”丢失，故障“秒” 切换
- 跨城市物理日志高速复制，最优可实现秒级RPO
- 中心级多数派一致性切换，避免中心级双主

其他预告

KES-AI时代的融合数据库架构



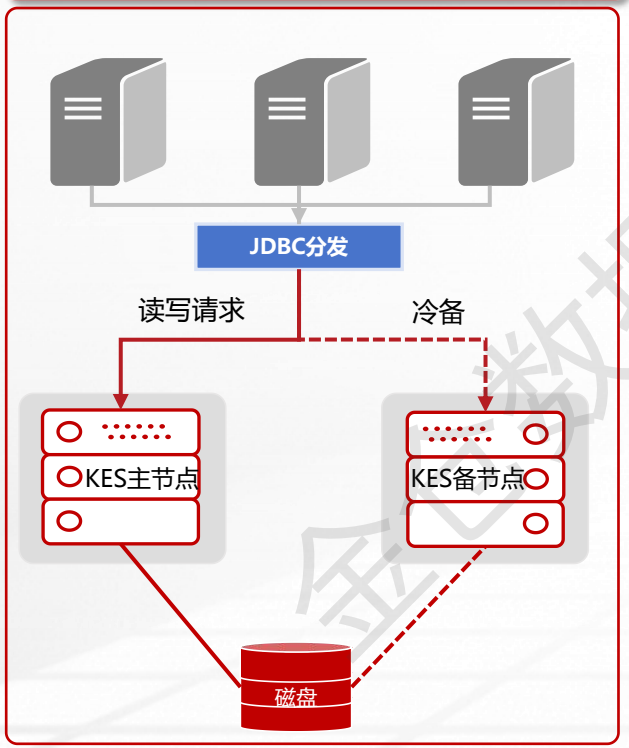
一体化层次	一体化目标
多语法体系一体化兼容	**平替：** “0”代码修改完成应用迁移
集中分布一体化架构	多集群架构**满足不同级别的**可用/**业务连续性、性能扩展性、成本需求**，最大化投资价值
多应用场景一体化处理	满足**企业级应用与AI创新等不同场景**的数据存算需求
多模数据一体化存储	**收敛技术栈**，降低应用复杂度和成本 减少库间数据同步，**降低同步开销**
开发运维一体化管理	**AI赋能开发运维：**基于大模型的开发与运维解决方案，显著提升工作效率

锚定长期技术方向，持续研发改进

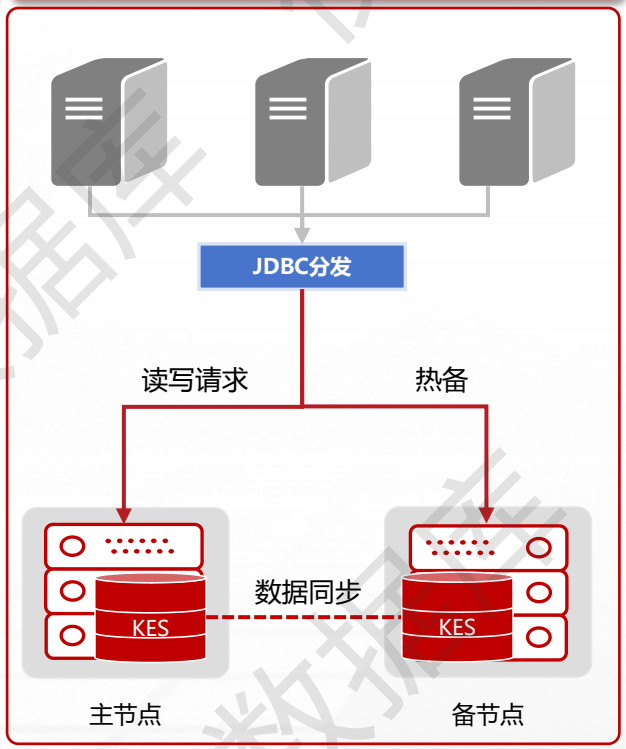
KES“集中式”集群架构

KINGBASE | 金仓社区

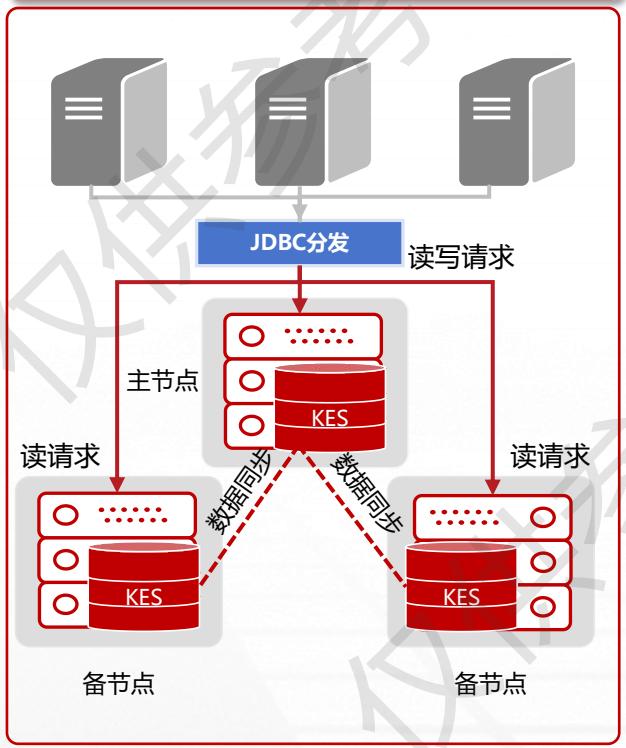
基于共享存储的主备方案（RAC One Node）



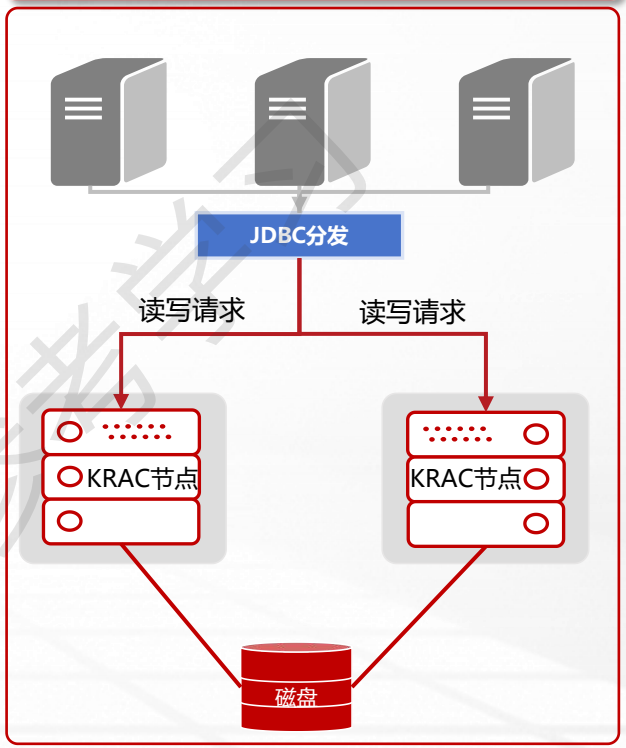
基于物理流复制的主备方案（RWC）



基于物理流复制的读写分离方案（RWC）



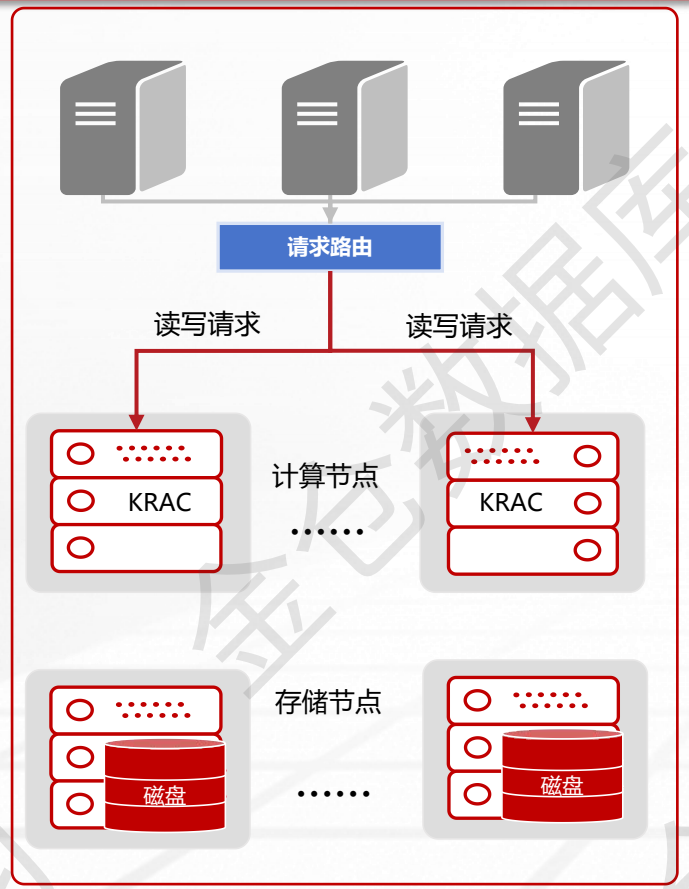
基于共享存储的多写方案（RAC）



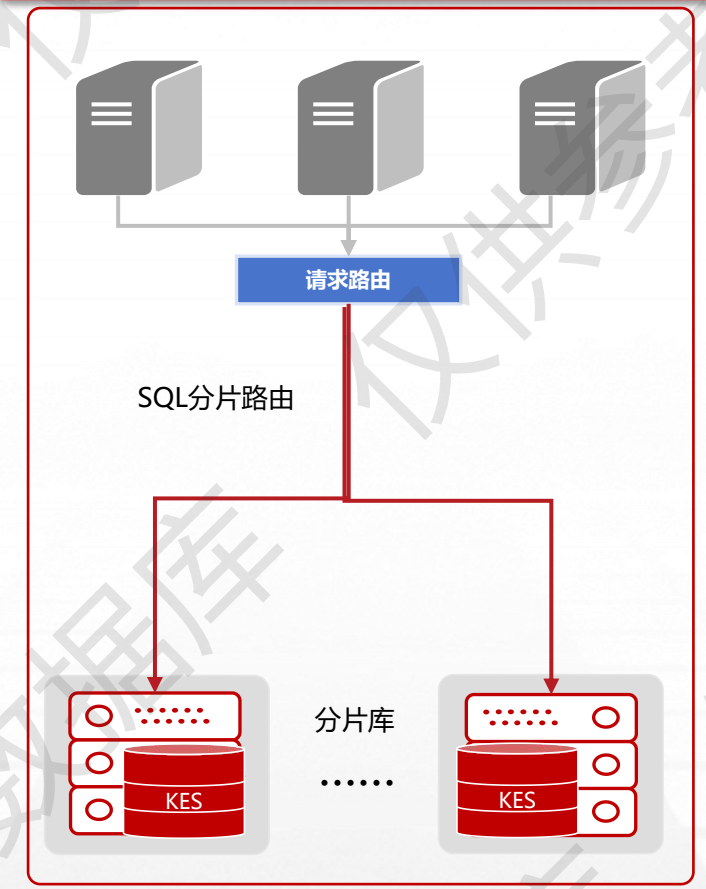
KES“分布式”集群架构

KINGBASE | 金仓社区

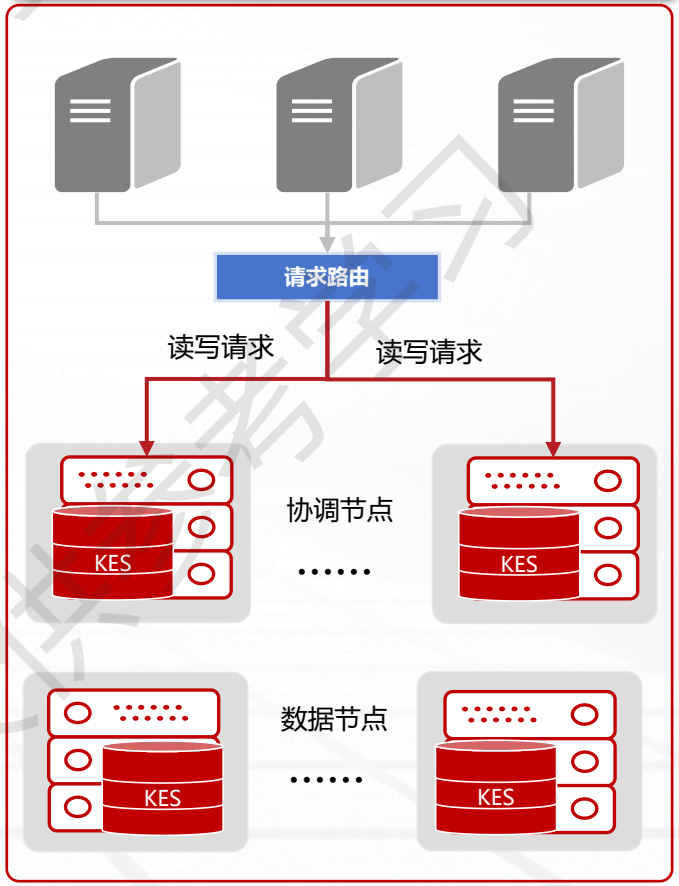
基于分布式存储的透明分布式方案（TDC）



基于分布式中间件的分布式方案（Sharding）



原生分布式方案（ADC）



多模数据一体化存储，一条SQL完成复杂检索

KINGBASE | 金仓社区

典型场景需求

查询某市过去30天内与雨天+分支道路+追尾形式类似的交通事故频发（至少3次）的5个热点区域

返回结果包含

- 事故数量
- 事故详情
- 最早和最后发生时间
- 与预期特征的相似度

关键数据模型

- **某市热点区域**：固定地理范围，以及细化区块/网格 -> GIS数据模型
- **雨天+分支道路+追尾形式**：相似度检索 -> 向量数据模型，按文本描述的语义相似度查询
- **区域内各事故详情**：文档聚合 -> 文档数据模型

```
WITH city_boundary AS (
    SELECT geom AS city_geom
    FROM areas
    WHERE name = '西安市'
    LIMIT 1
), filtered_accidents AS (
    SELECT
        a.accident_id,
        a.accident_time,
        a.location,
        a.details,
        a.embedding,
        ar.area_id,
        ar.name AS area_name,
        a.embedding <=> '[0.12037, 0.49821, 0.42626, ……]'::vector(768) AS vector_distance
    FROM accidents a
    CROSS JOIN city_boundary
    JOIN areas ar ON ST_Contains(ar.geom, a.location)
    WHERE ST_Within(a.location, city_boundary.city_geom)
    AND a.accident_time >= NOW() - INTERVAL '30 days'
    AND a.embedding <=> '[0.12037, 0.49821, 0.42626, ……] '::vector(768) < 0.5
)
SELECT
    fa.area_id,
    fa.area_name,
    COUNT(*) AS accident_count,
    json_agg(fa.details) AS aggregated_details,
    MIN(fa.accident_time) AS first_accident,
    MAX(fa.accident_time) AS last_accident,
    AVG(fa.vector_distance) AS avg_vector_distance
FROM filtered_accidents fa
GROUP BY fa.area_id, fa.area_name
HAVING COUNT(*) > 3
ORDER BY accident_count DESC
LIMIT 5;
```

说明：

- 事故的描述文字通过嵌入模型生成向量（本例中为 768 维），保存在表 a 的 embedding 字段中；
- 查询输入的文本“雨天在分支道路上追尾”，也通过上述模型生成嵌入向量（内容过长，以 [0.12037, 0.49821, 0.42626, ……] 示意）；

GIS计算
明确事故所在区域

向量搜索
事故特征与查询特征的余弦距离

文档处理
分区域聚合事故详情清单

分组排序
分组计算热点区域排序

KINGBASE | 金仓社区

THANKS

成为世界卓越的数据库产品与服务提供商