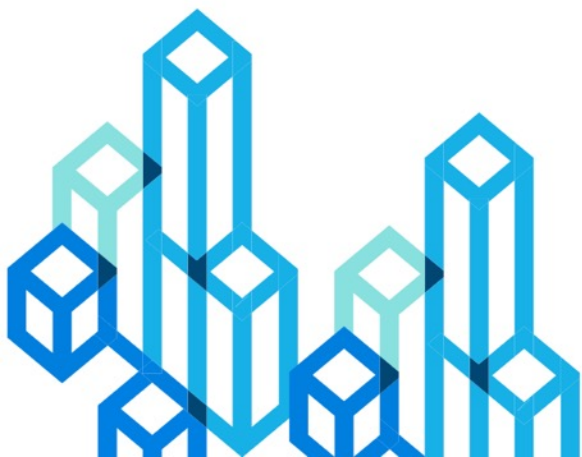


从DolphinDB浅谈时序数据库设计

周信静

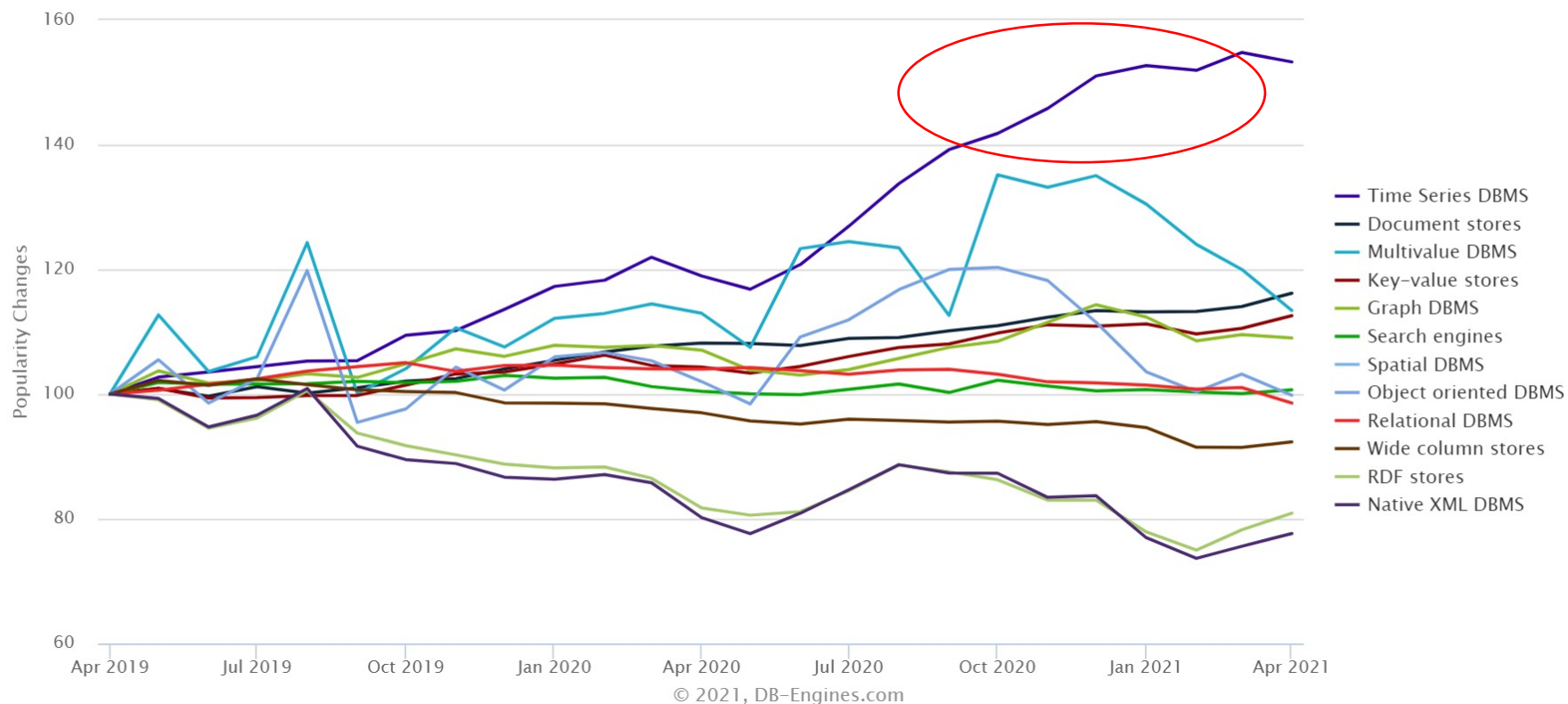


- 本硕毕业于杭州电子科技大学和浙江大学
- 原腾讯云数据库CynosDB/CDB研发工程师
- 现DolphinDB存储引擎研发负责人

- 时序数据特点和挑战
- DolphinDB
 - TSDB存储引擎设计
 - 为什么时序数据库需要事务以及怎么做
- 总结

时序数据库是近年来增长最快的数据库类别

Trend of the last 24 months



金融市场



工业机器



能源




通信




交通运输




- Tags
 - 实体，数据源
 - e.g., machine, device, stock symbol ...
- Metrics
 - 实体随时间变化的属性
 - e.g., utility, temperature, pressure, price
- Timestamp
- 举例：能源领域
 - site, generator group, machine, temperature, pressure, ts



Tags



Metrics



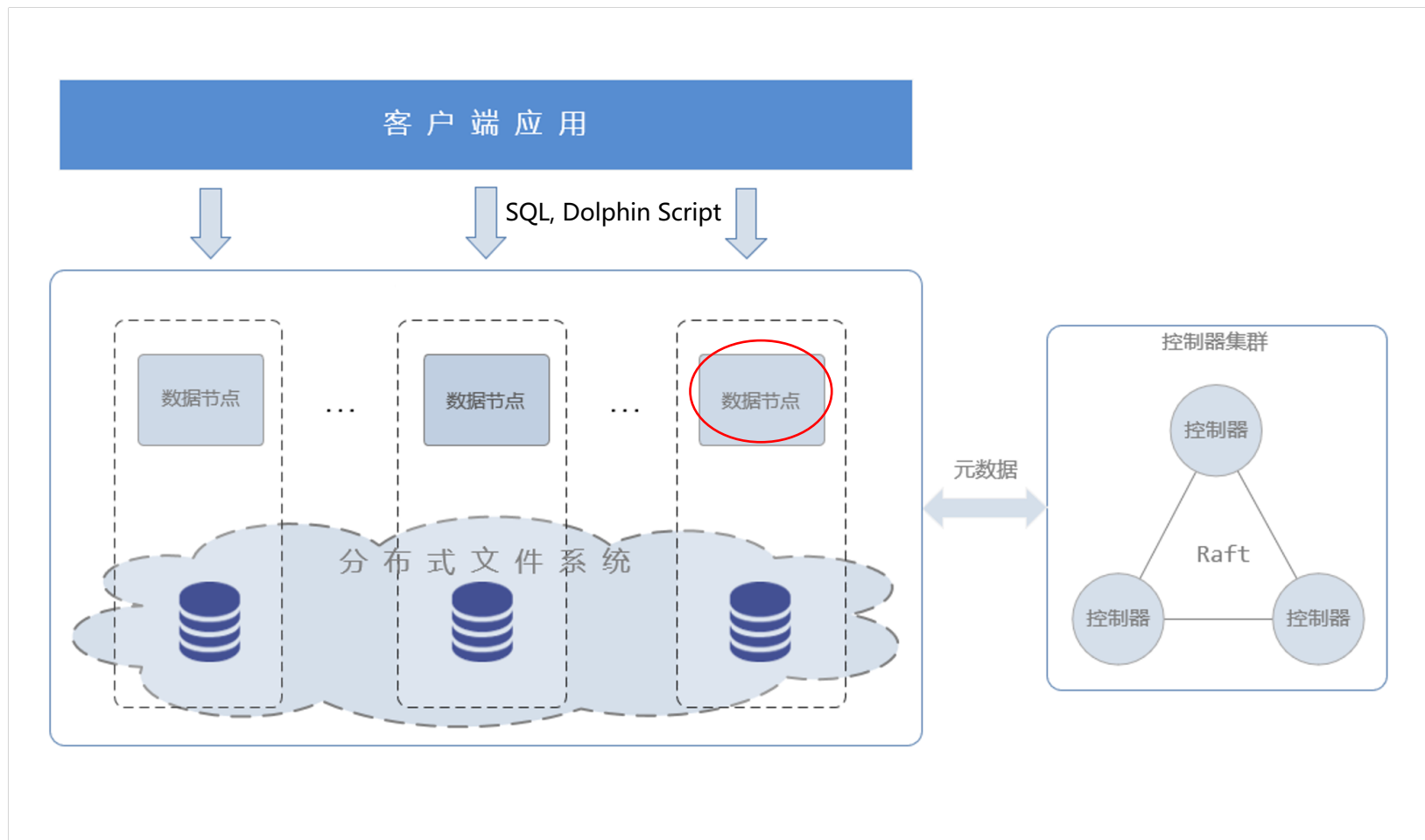
Timestamp

- 1. 写入吞吐要求高
 - 浙江省电网千万电表 => 千万写入每秒
 - 远高于OLTP负载
 - 周期性持续性批量写入
 - 少量更新, 批量删除
- 2. 数据量大
 - e.g., 波音787每次飞行产生0.5TB时序数据
 - Tag维度组合爆炸
 - 数据价值密度低
 - 期望存储成本低 => 大容量低价HDD
- 3. 典型查询负载：对时间序列毫秒级别点查询、窗口聚合
 - e.g., 过去一个小时，某个机器温度均值是多少？

- MySQL, PostgreSQL: 几十万写入/s
 - TimescaleDB
- 问题
 - B+Tree容易退化，碎片问题，无法维持高写入吞吐
 - 无法scale到大数据量

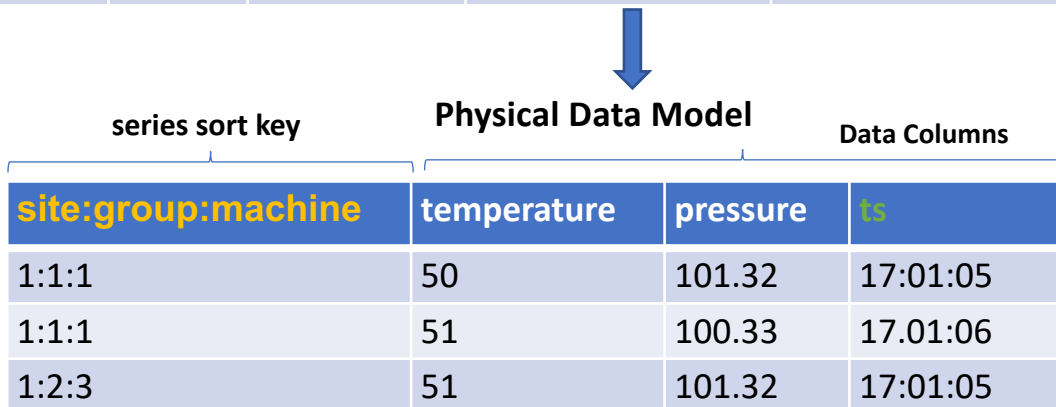
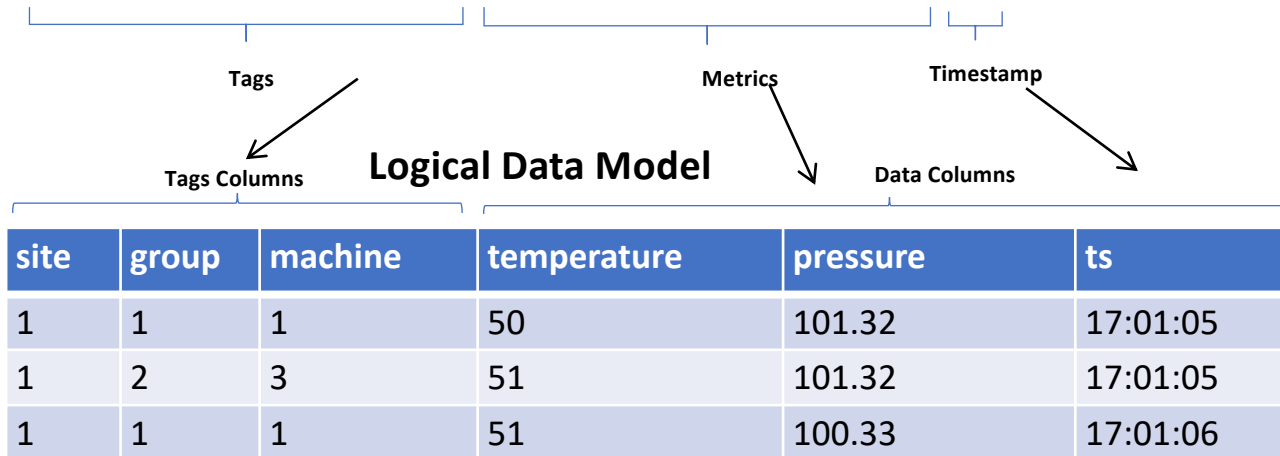
- Key = tags+metric+timestamp
- Value = metric value
- OpenTSDB(HBase), Cassandra
- 问题
 - Key冗余部分太多，存储成本高
 - key数量巨大，性能低下

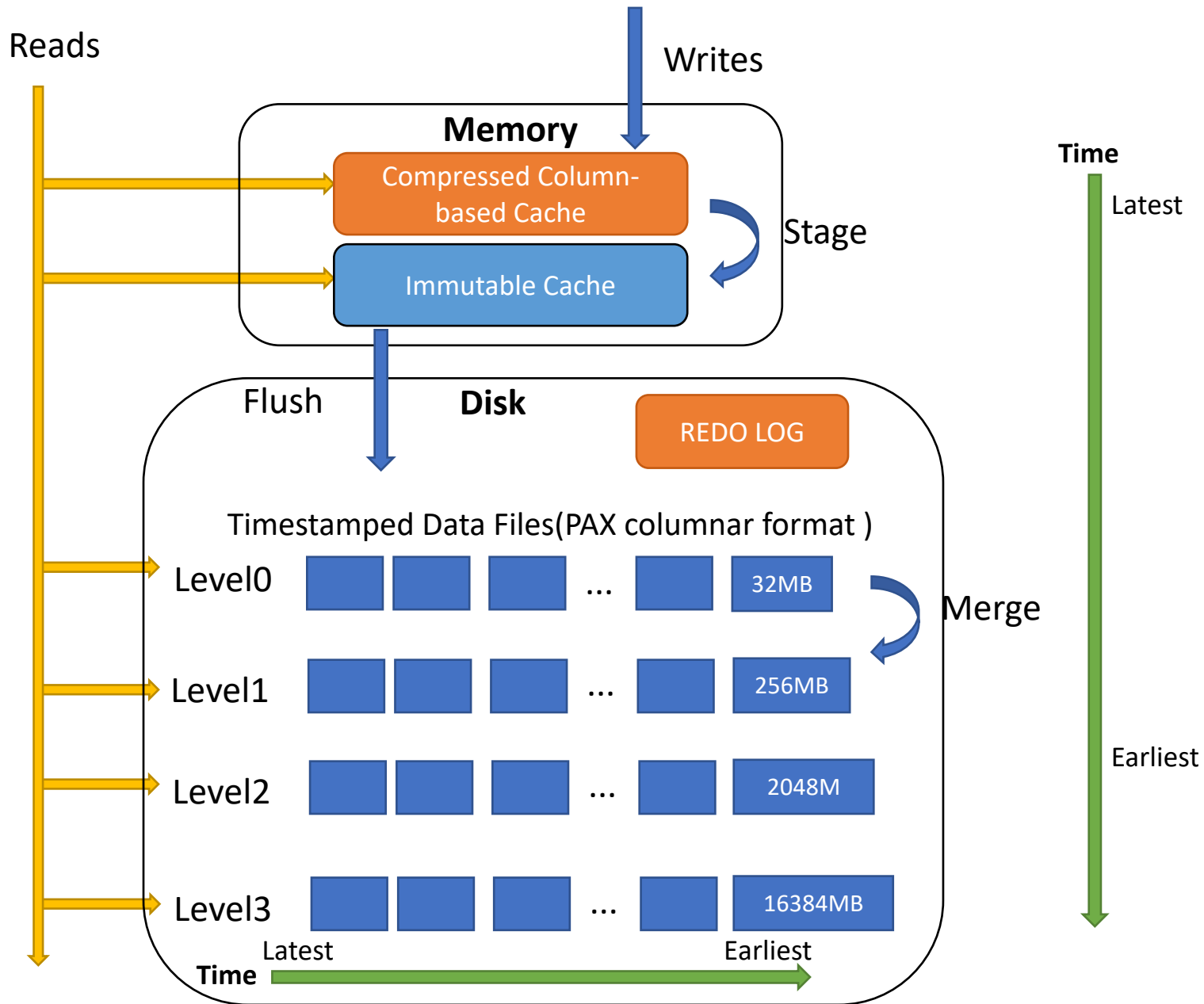
时序数据需要特殊数据库支持!

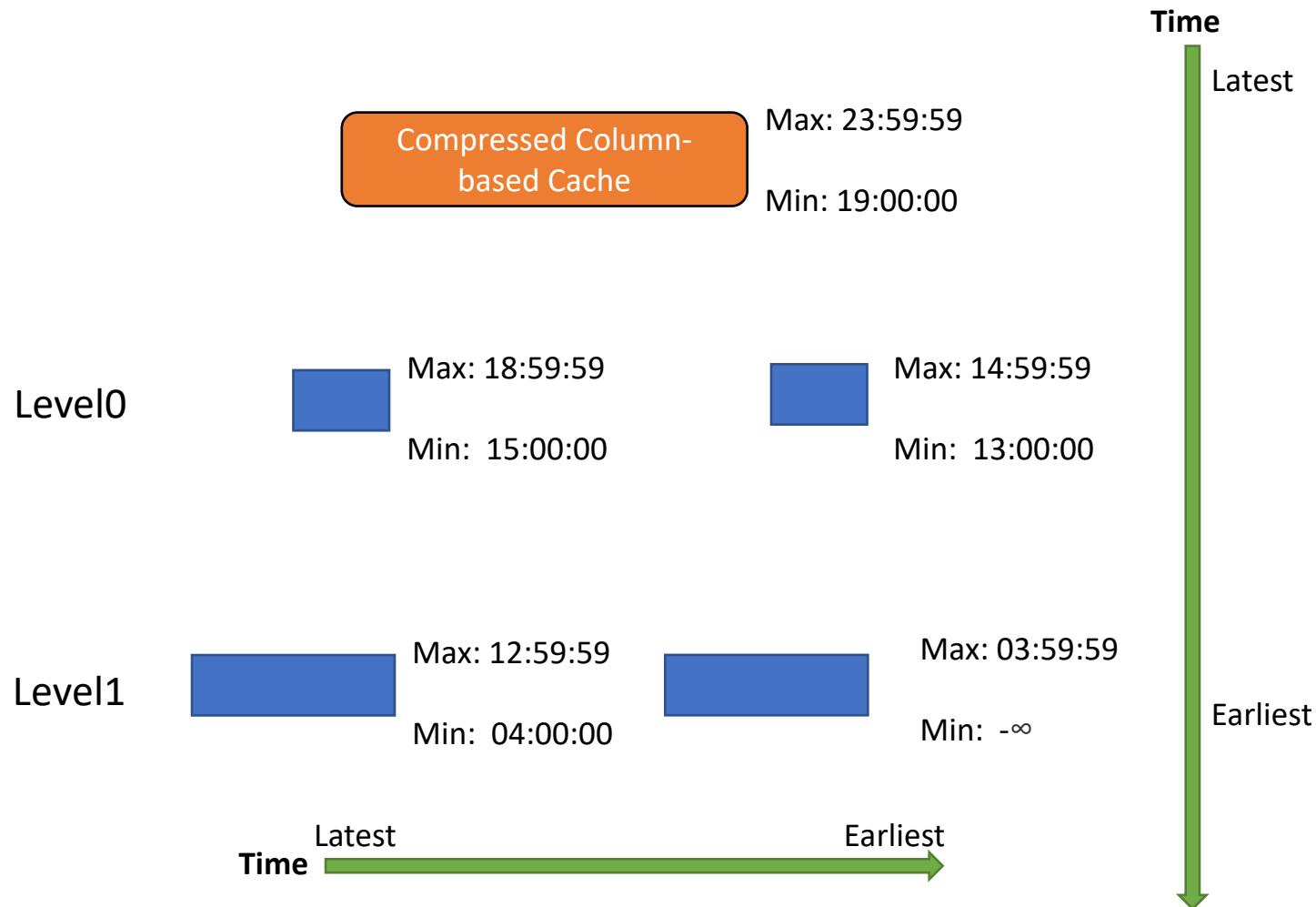


- 设计目标
 - 写入高吞吐
 - 单个时序序列定位快
 - 支持高速时序数据分析
 - 低存储成本

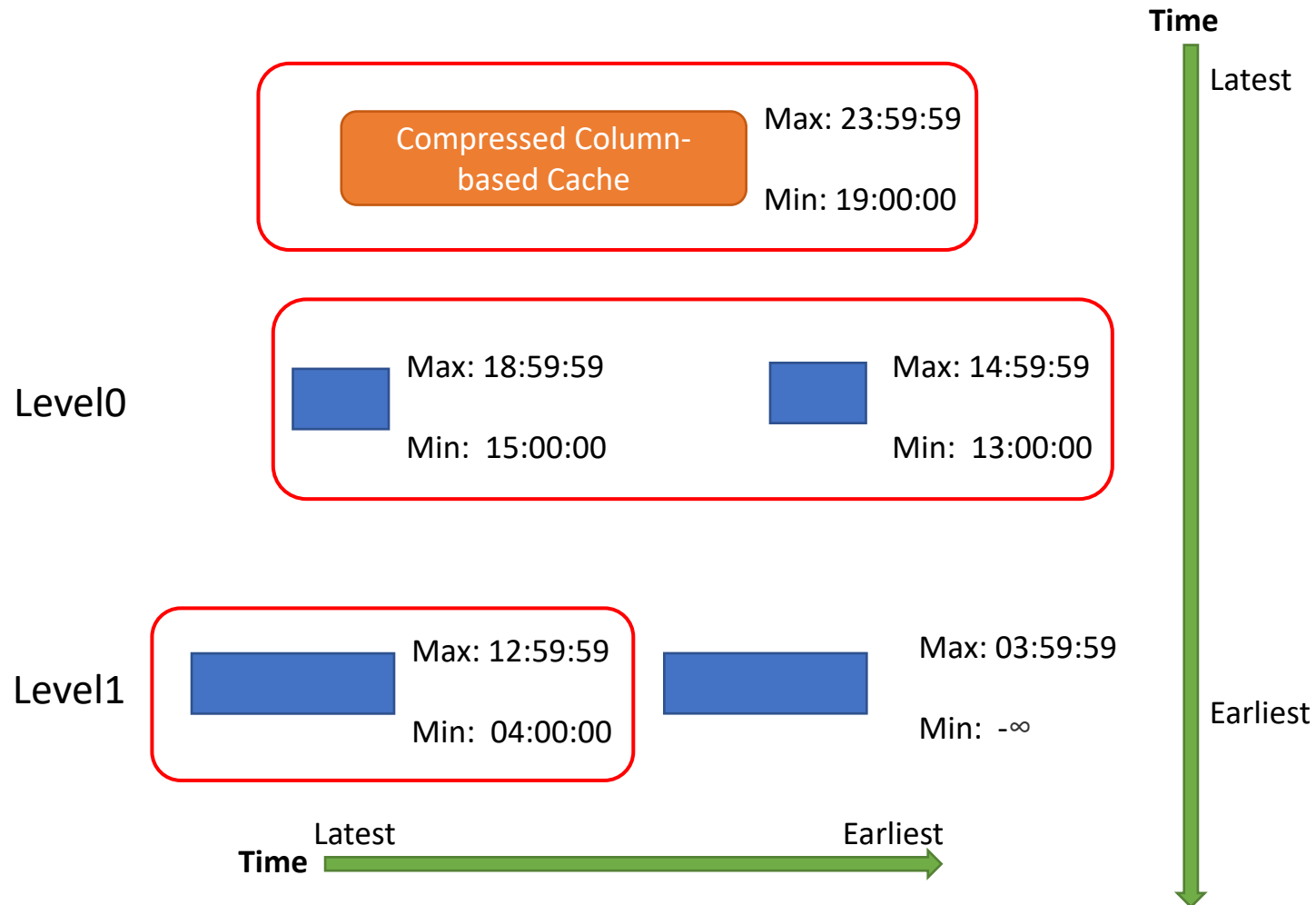
site, group, machine, temperature, pressure, ts

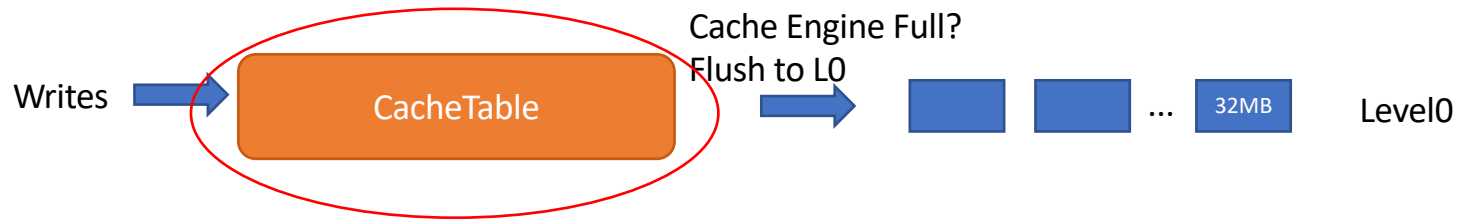


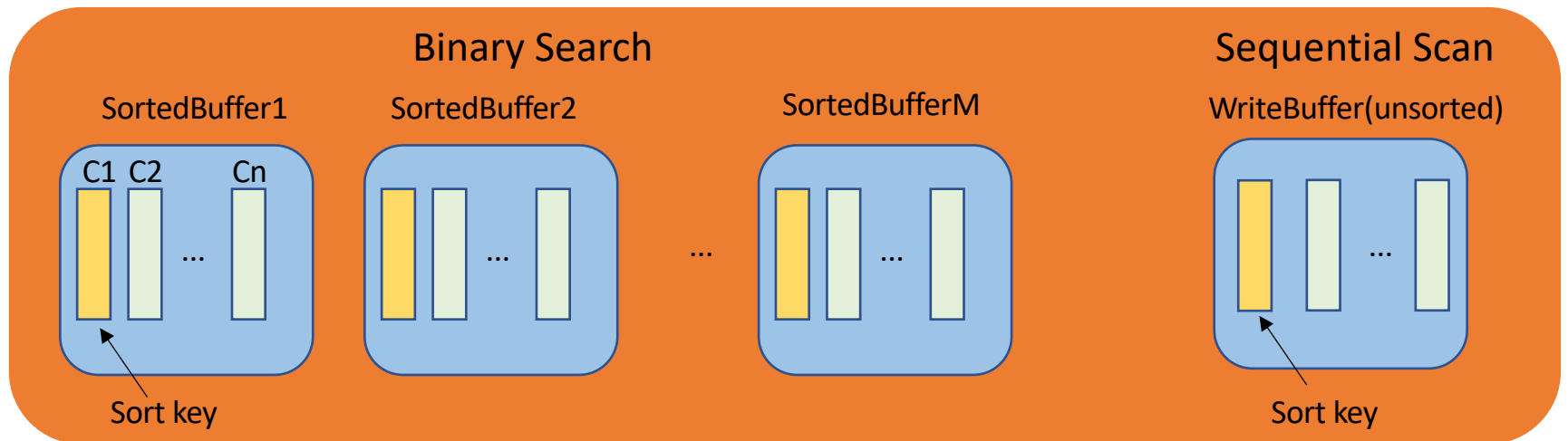


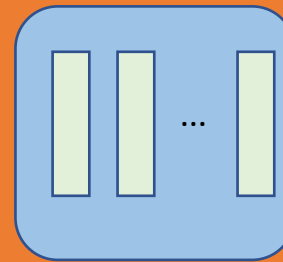


select * from t where site=1 and group=1 and machine = 1
and ts between 05:00:00 and 20:00:00



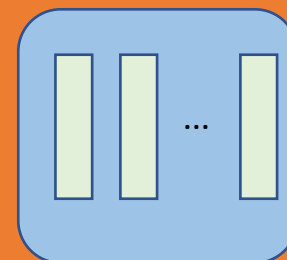






← 写入快速append到
WriteBuffer尾部

WriteBuffer行数超过threshold,
转成一个sorted buffer,
清空WriteBuffer

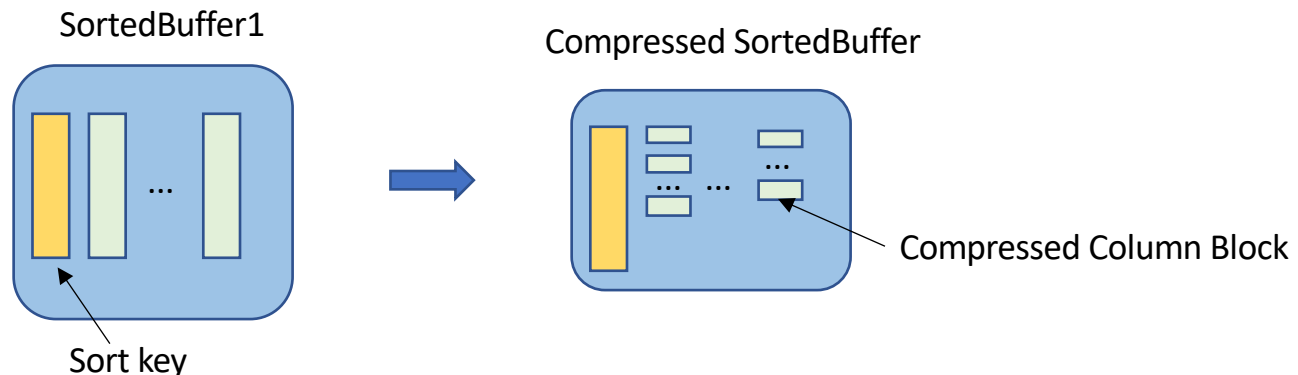


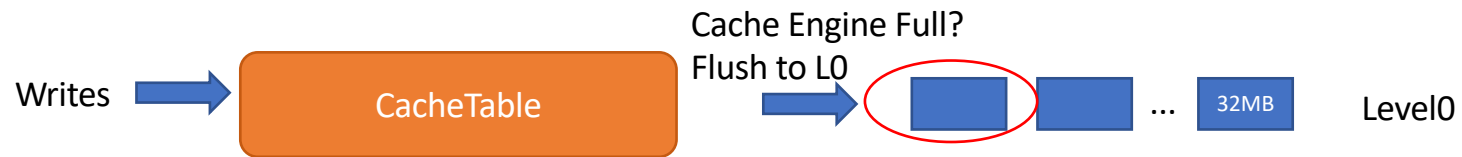
- 观察

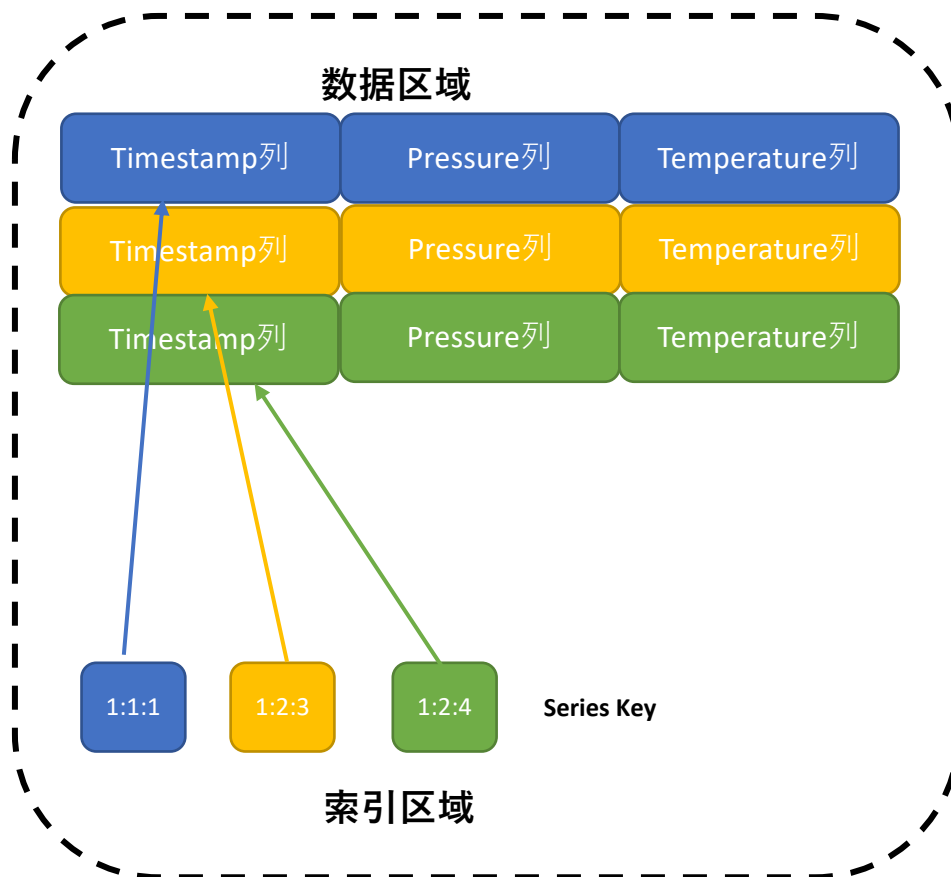
- HDD 4KB 随机读延迟:**5-15ms**, Flash SSD 4KB随机读延迟: **100-200us**
- 4KB snappy decompression: **1~2us**
 - 相比于访问HDD/SSD, 在内存中解压更加划算
- Sort Buffer不变化+列存+时序数据 → 高度可压缩
- 时间局部性: 时间越近的数据越有可能被访问

- 优化

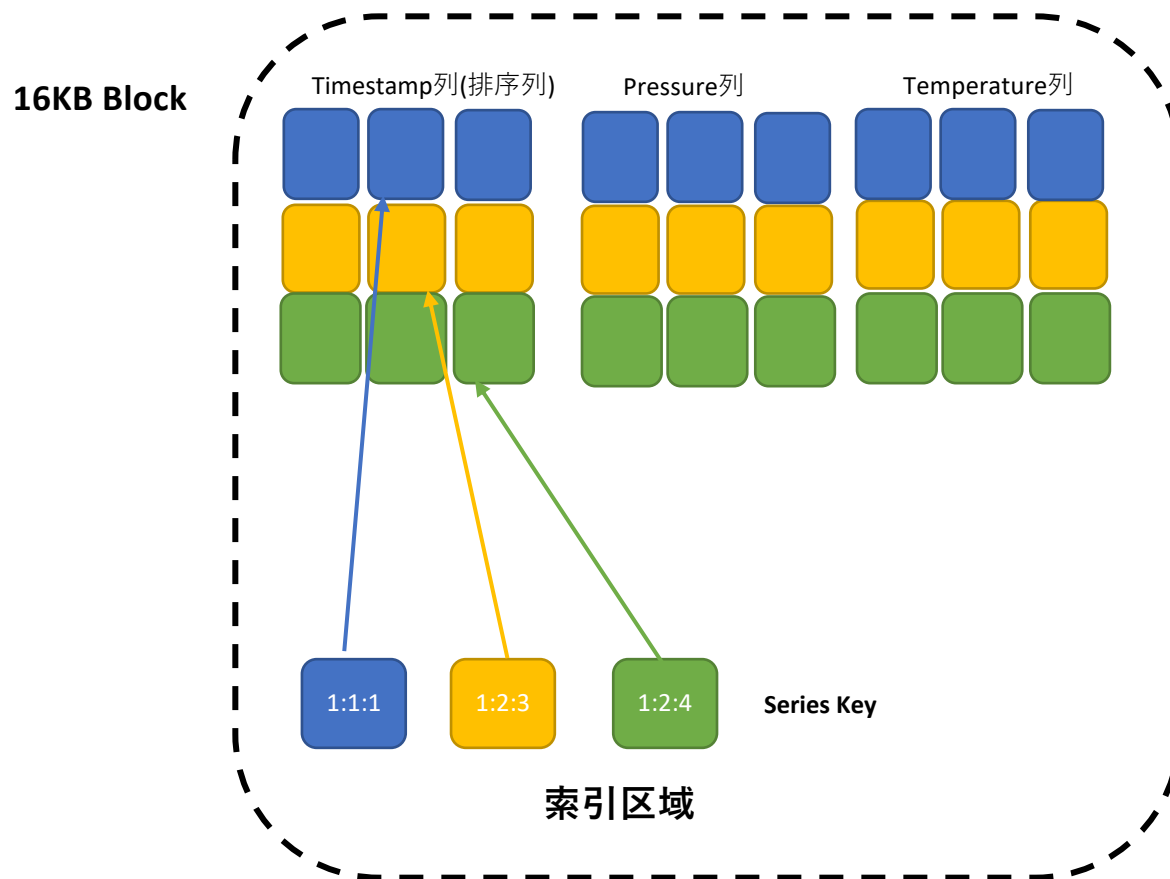
- 压缩sorted buffer, 缓存更多数据在Cache中, 减少IO
- 压缩比20%, 多缓存~5x数据在DRAM



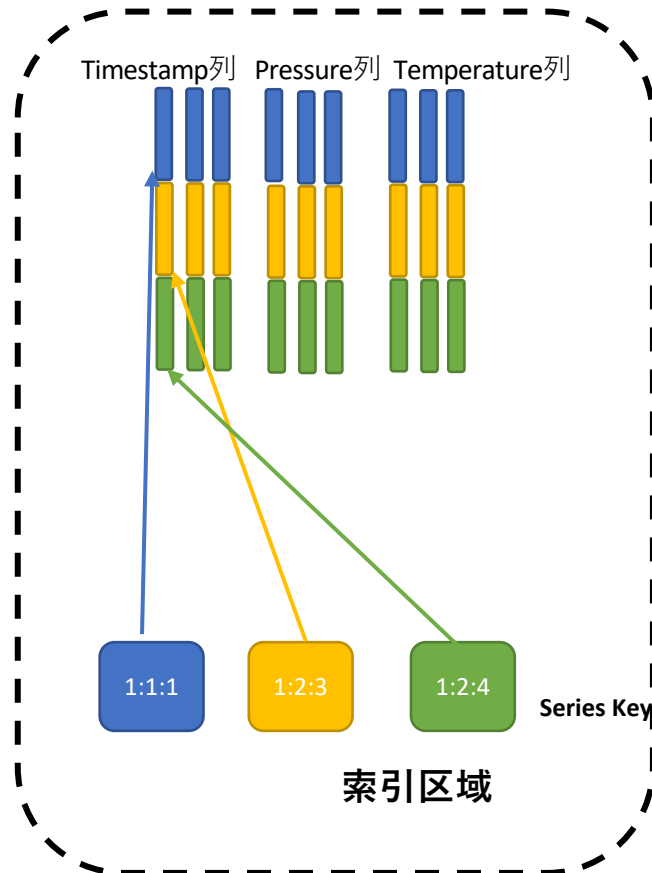




一个序列的Tags只存储一次在索引区，降低Tags冗余度



数据块按时间列排序, 过滤不需要的block



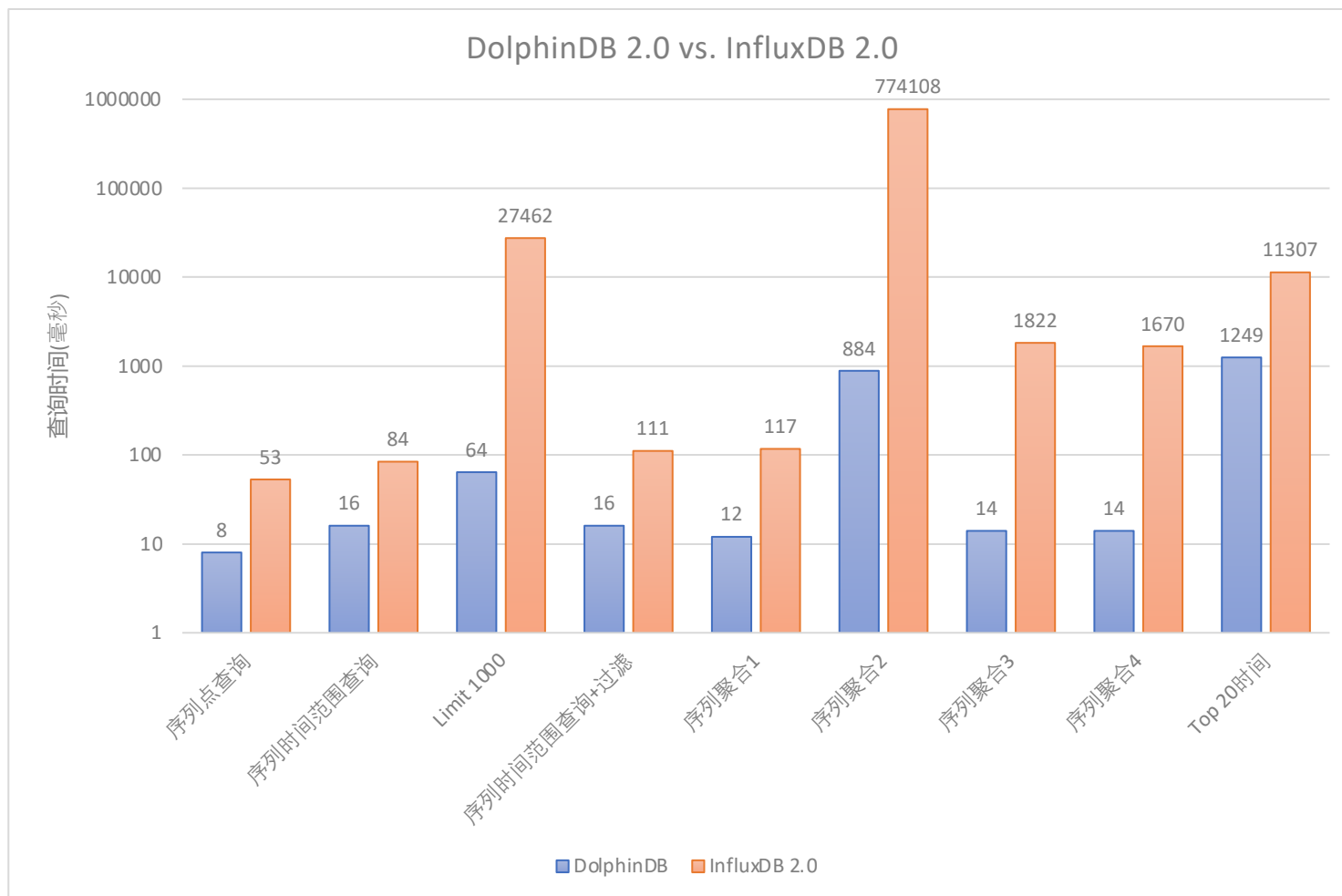
- 压缩算法

- Dictionary Encoding
- Delta of Delta
- LZ4

- Series级别预聚合 - zonemap
 - SUM/MIN/MAX/COUNT
- 存储在文件索引区域
- `SELECT avg(pressure) FROM devices WHERE site=xxx and group=yyy`

- 列式向量化执行引擎
- 分区级并行查询
 - Scan
 - Sorting
 - Join
 - GroupBy

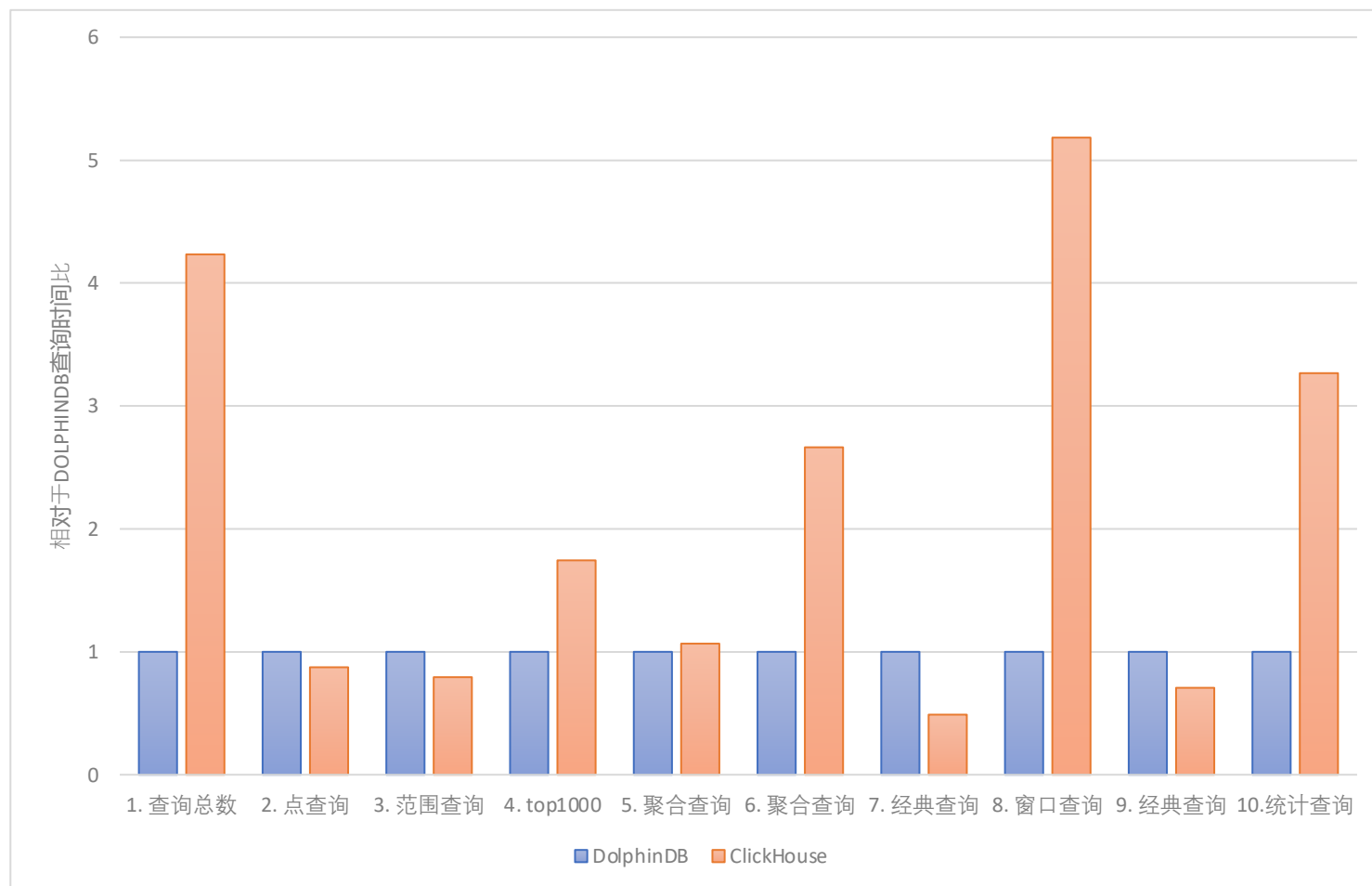
- 测评数据集¹: 传感器信息
- 导入吞吐: DolphinDB 1M rows/s, InfluxDB 14000 rows/s
- 压缩比: DolphinDB 2.7, InfluxDB 2.9



1. <https://docs.timescale.com/v1.1/tutorials/other-sample-datasets>

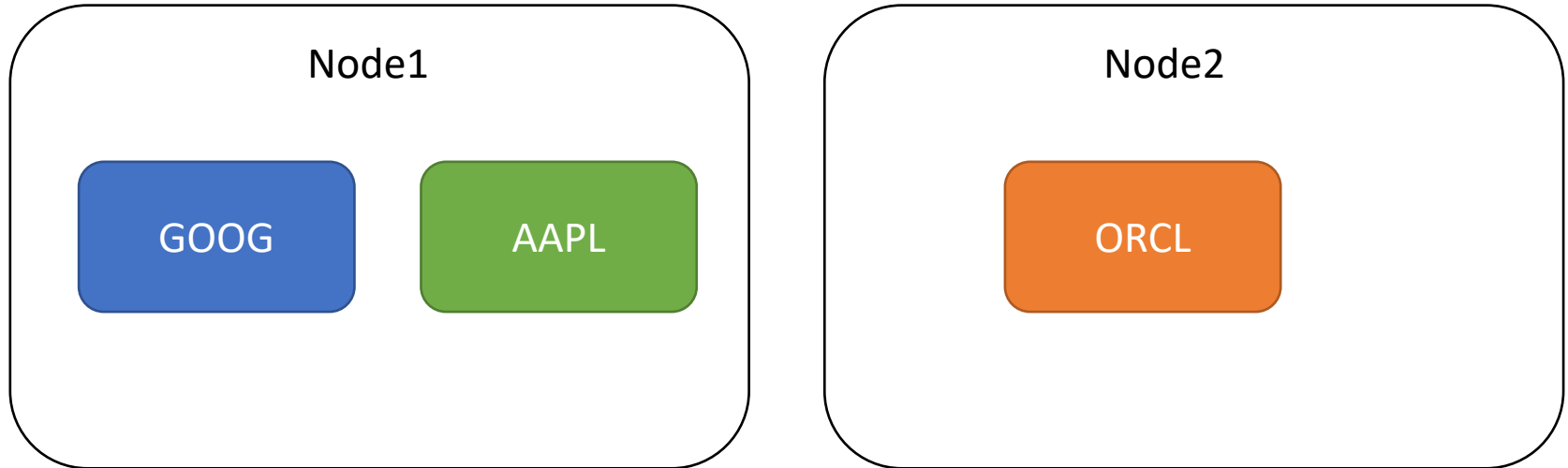
DolphinDB 2.0 vs. ClickHouse 21.7

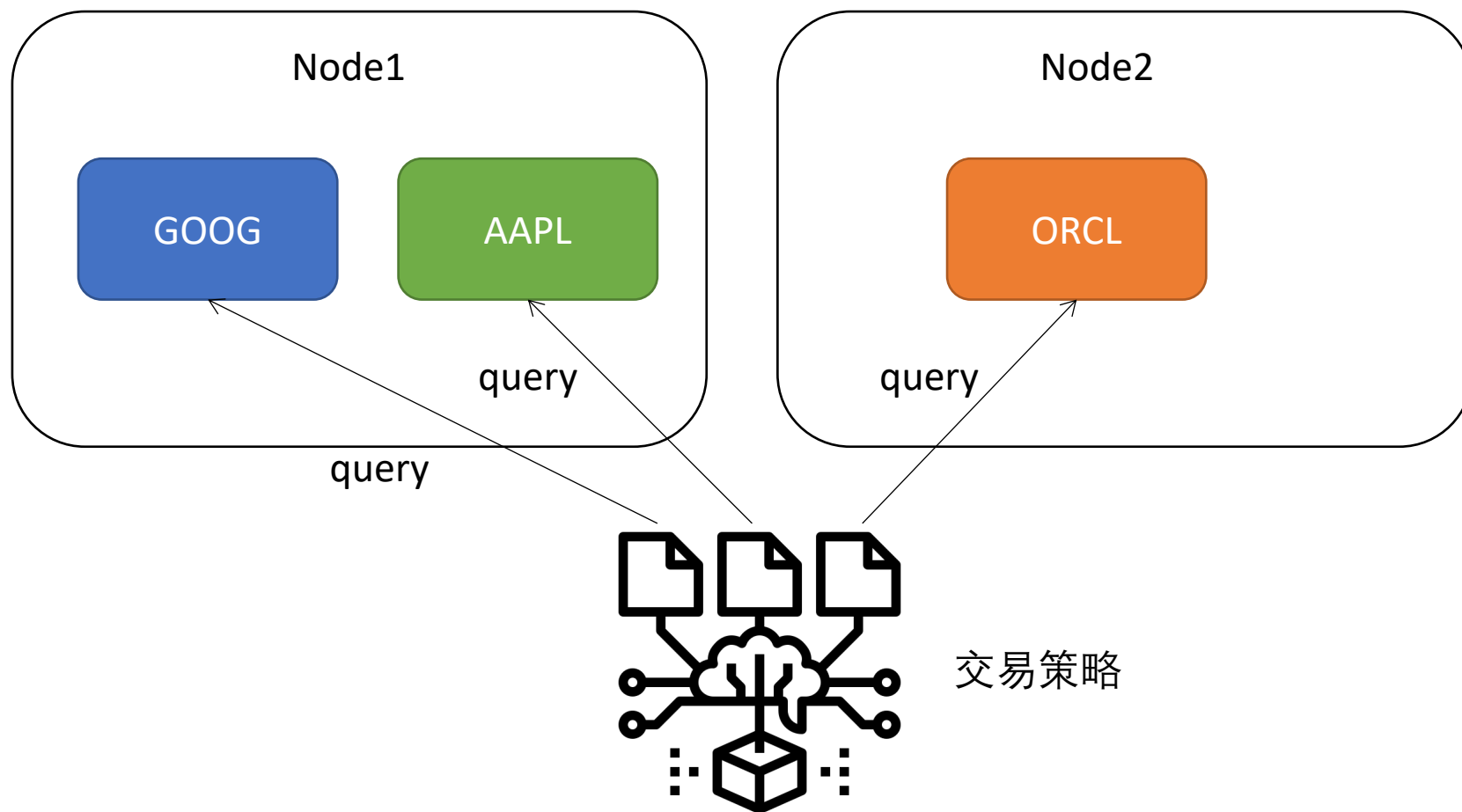
- 测评数据集: NYSE Trade and Quote(335GB)
- 导入吞吐: DolphinDB 70MB/s, ClickHouse 42MB/s

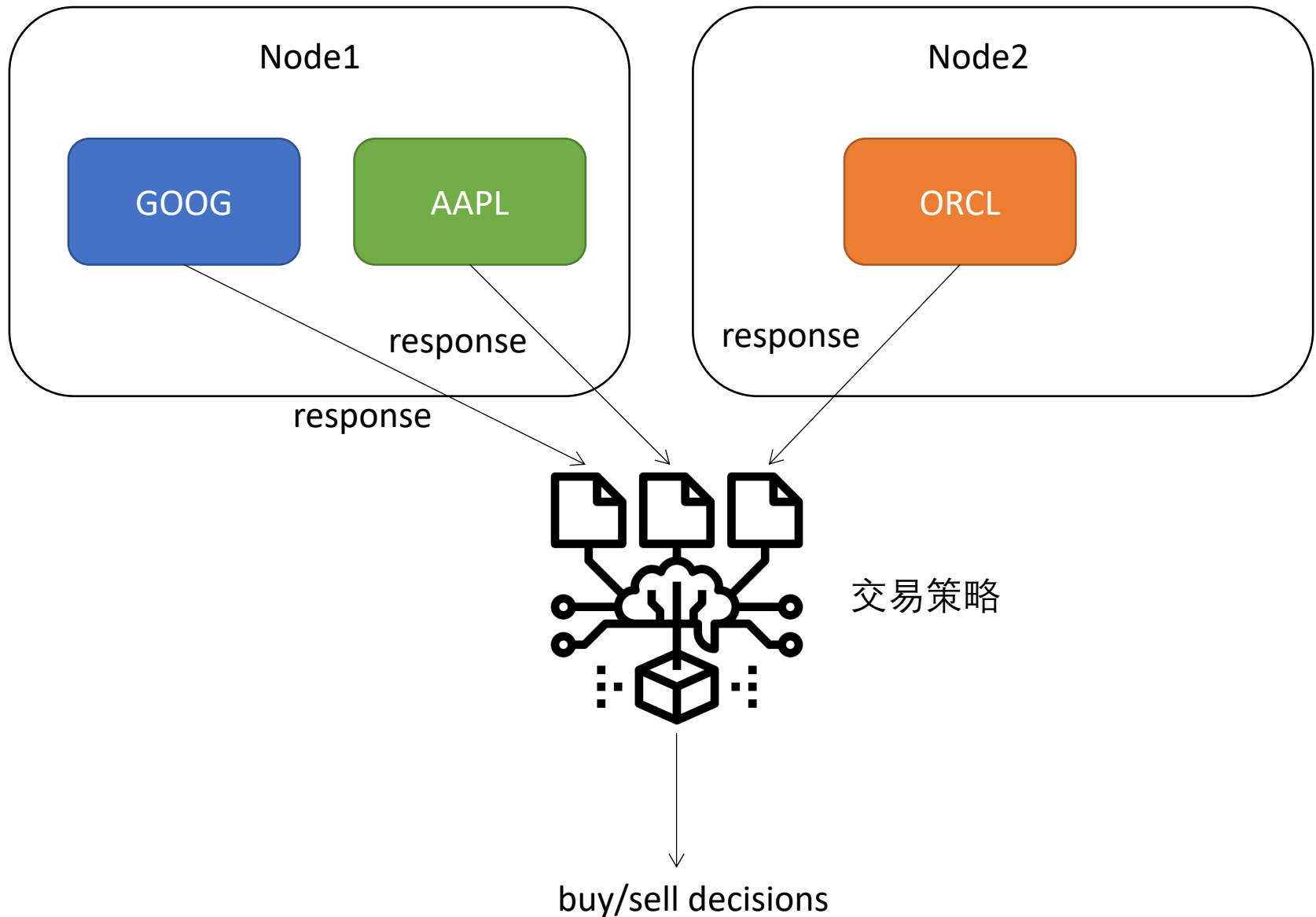


- 数据量巨大，单机能力不足，需要分布式存储和计算
- 数据分区存储在多个节点上
 - 按Tag分区
 - 按时间分区
 - daily/hourly ...
- 高可用
 - 分区需要多副本
- 分布式环境下失败并不罕见，数据一致性很重要
 - 机器宕机
 - 磁盘损坏
 - 软件bug

- 例子: 股票交易(symbol, time, price)







股票快照数据(3秒一批)

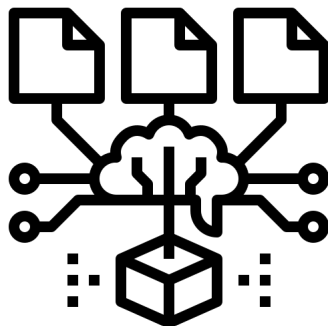
Node1

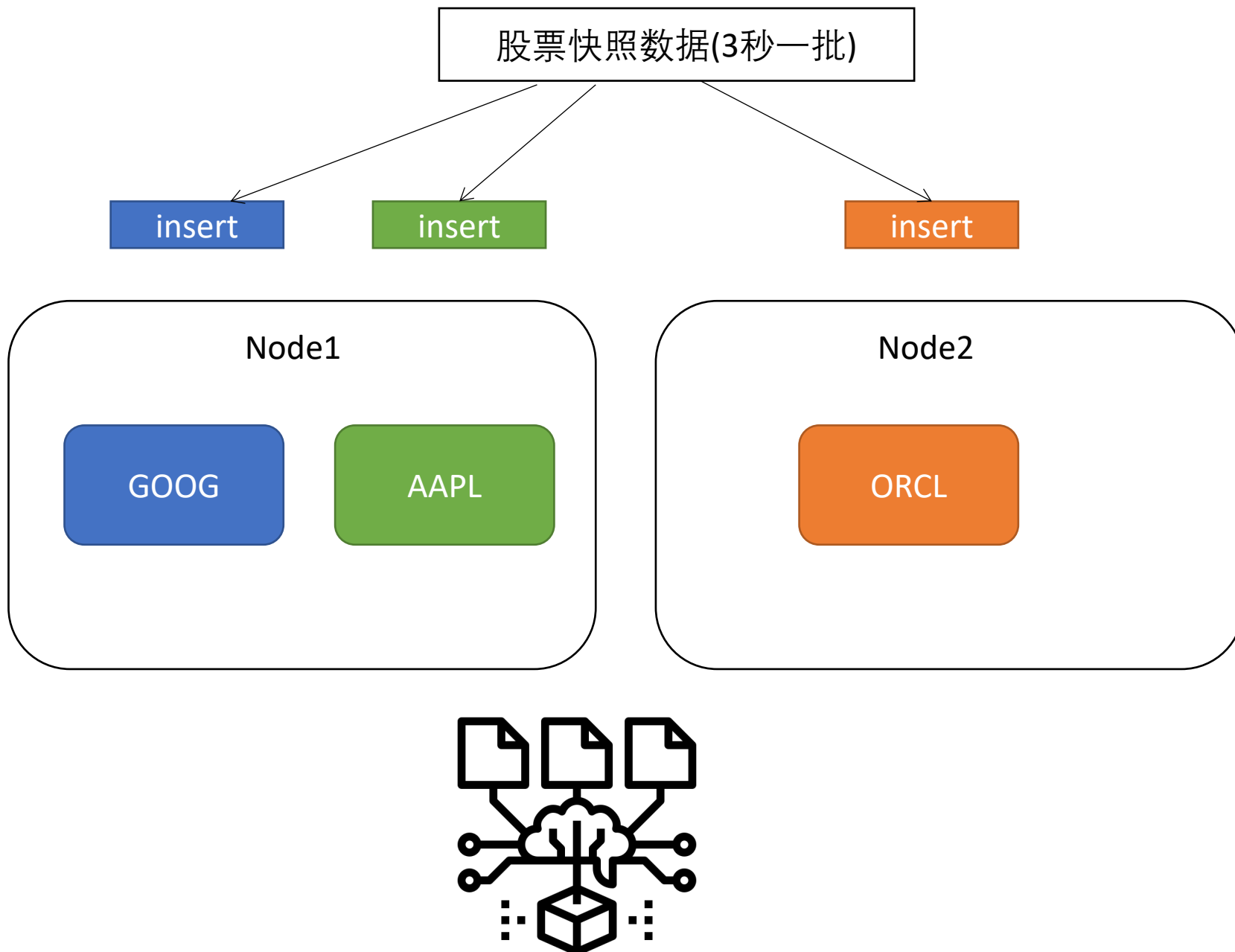
GOOG

AAPL

Node2

ORCL





Network delayed / Queuing

insert

Node1

insert

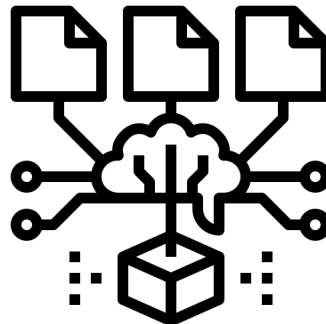
GOOG

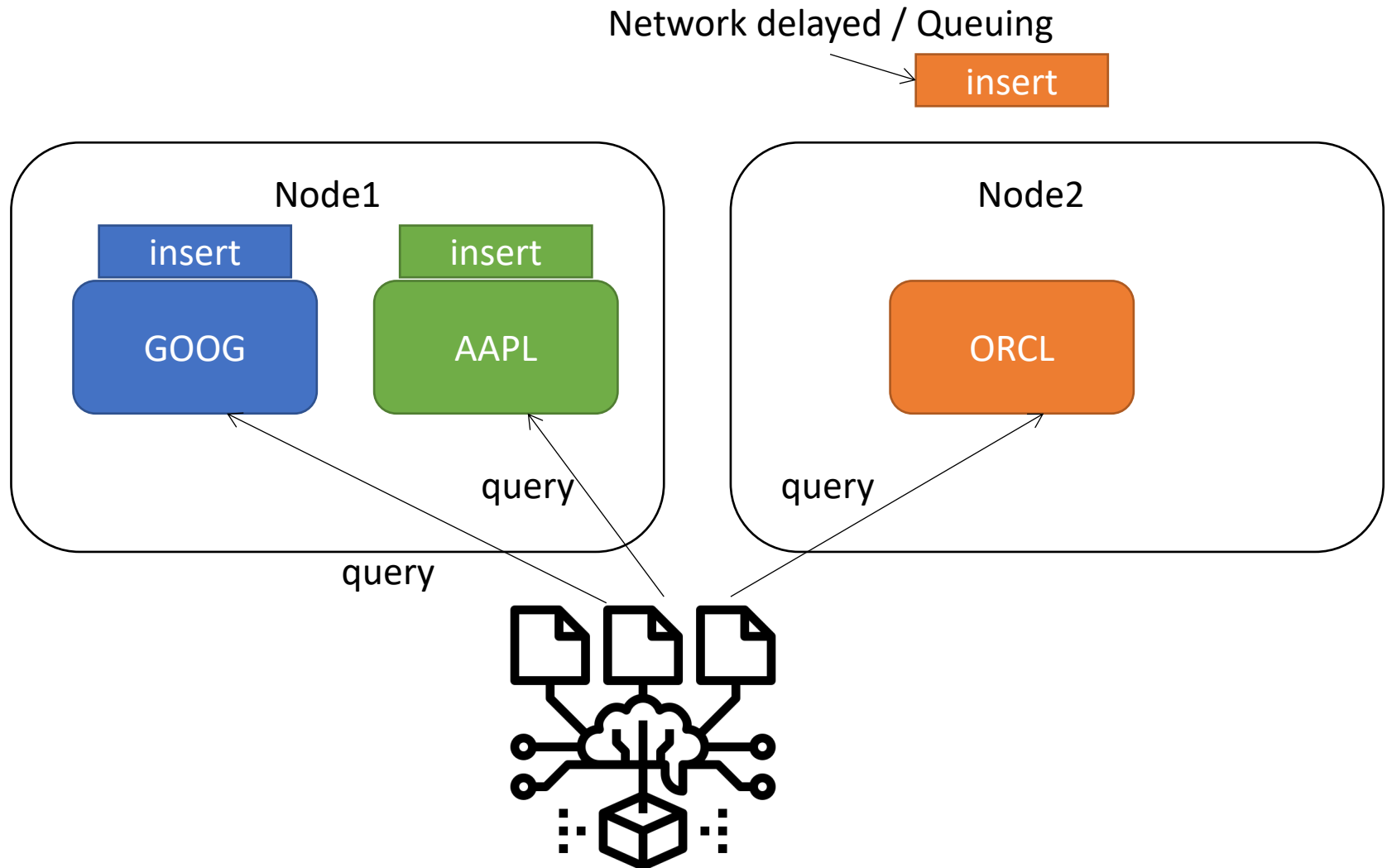
insert

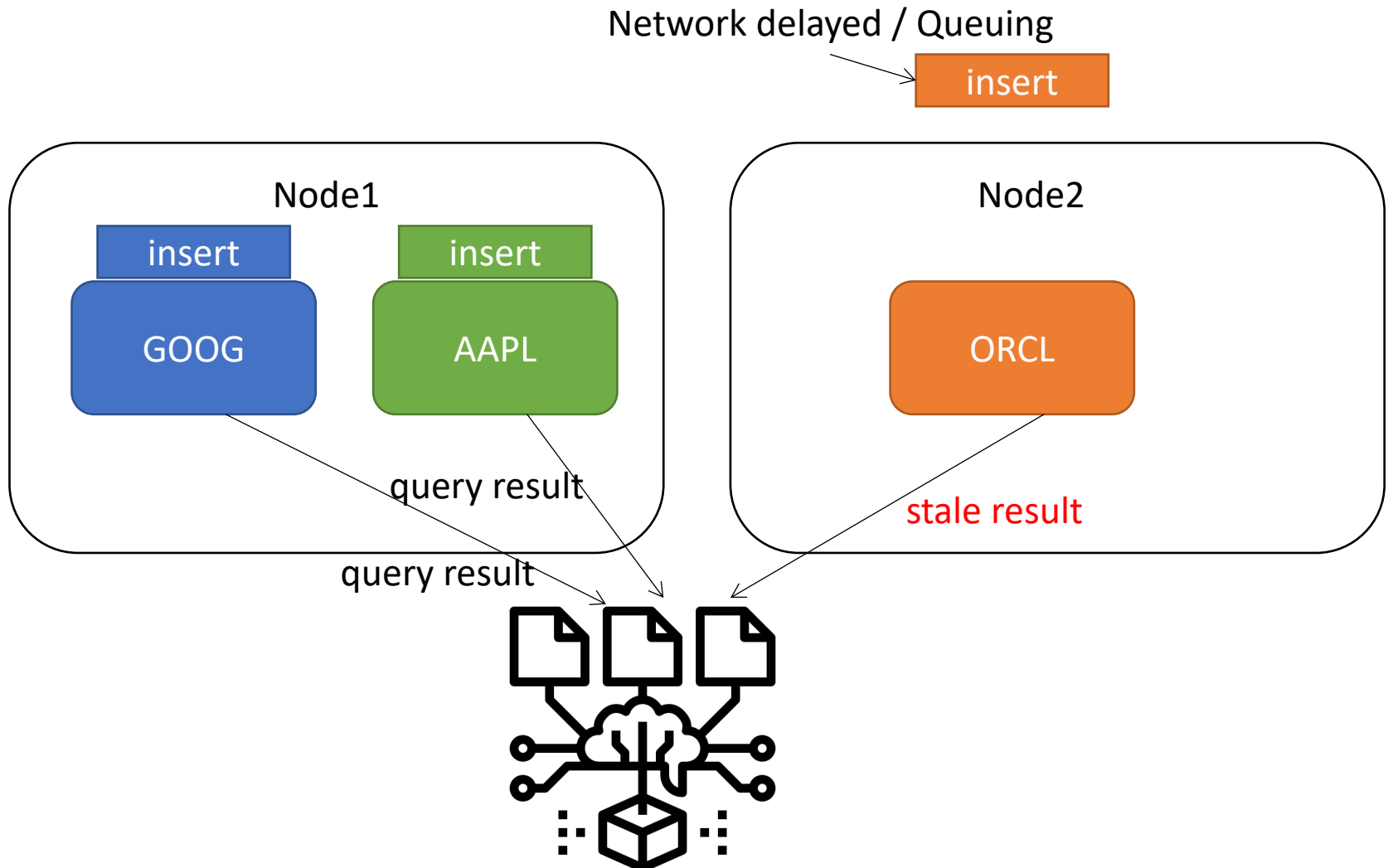
AAPL

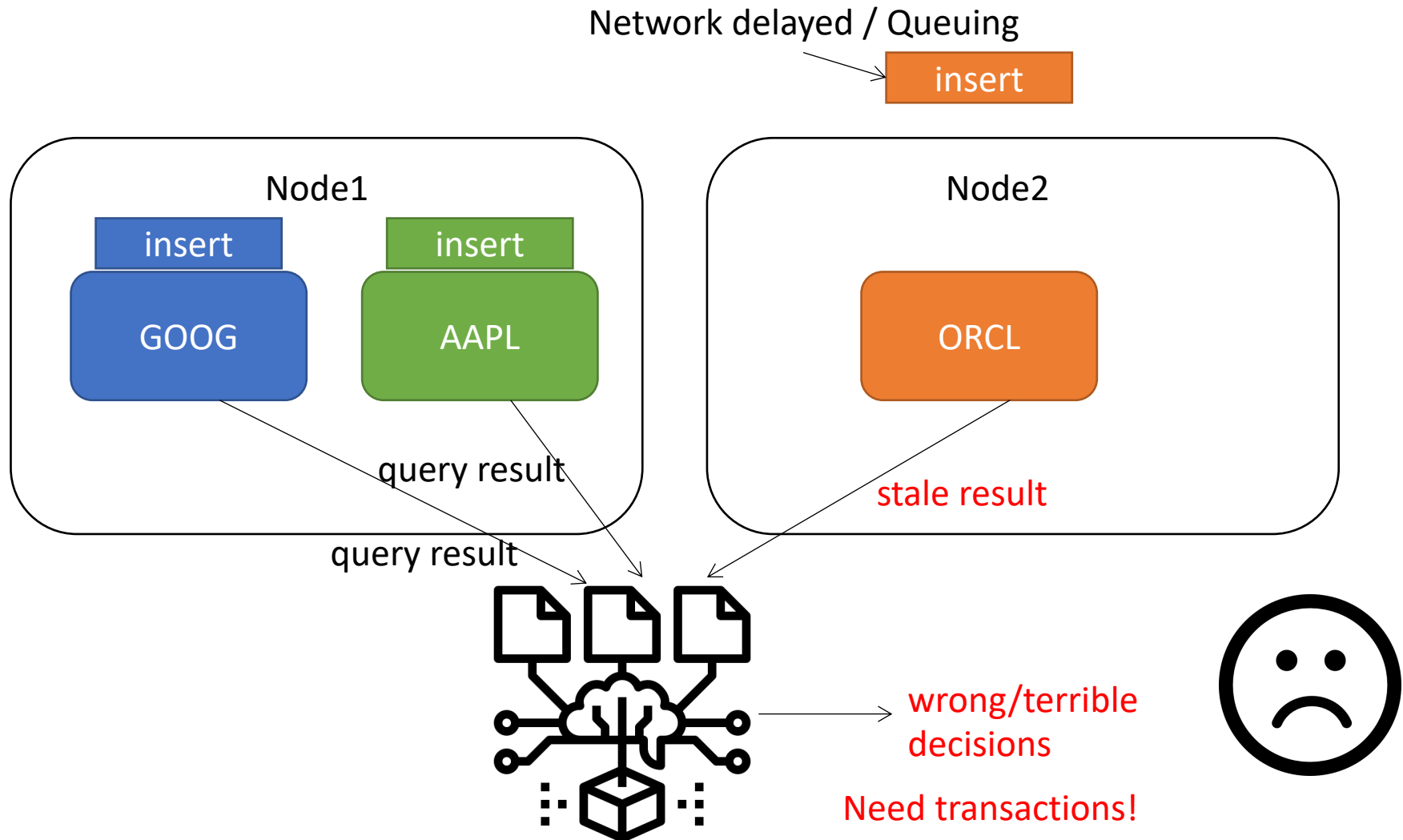
Node2

ORCL

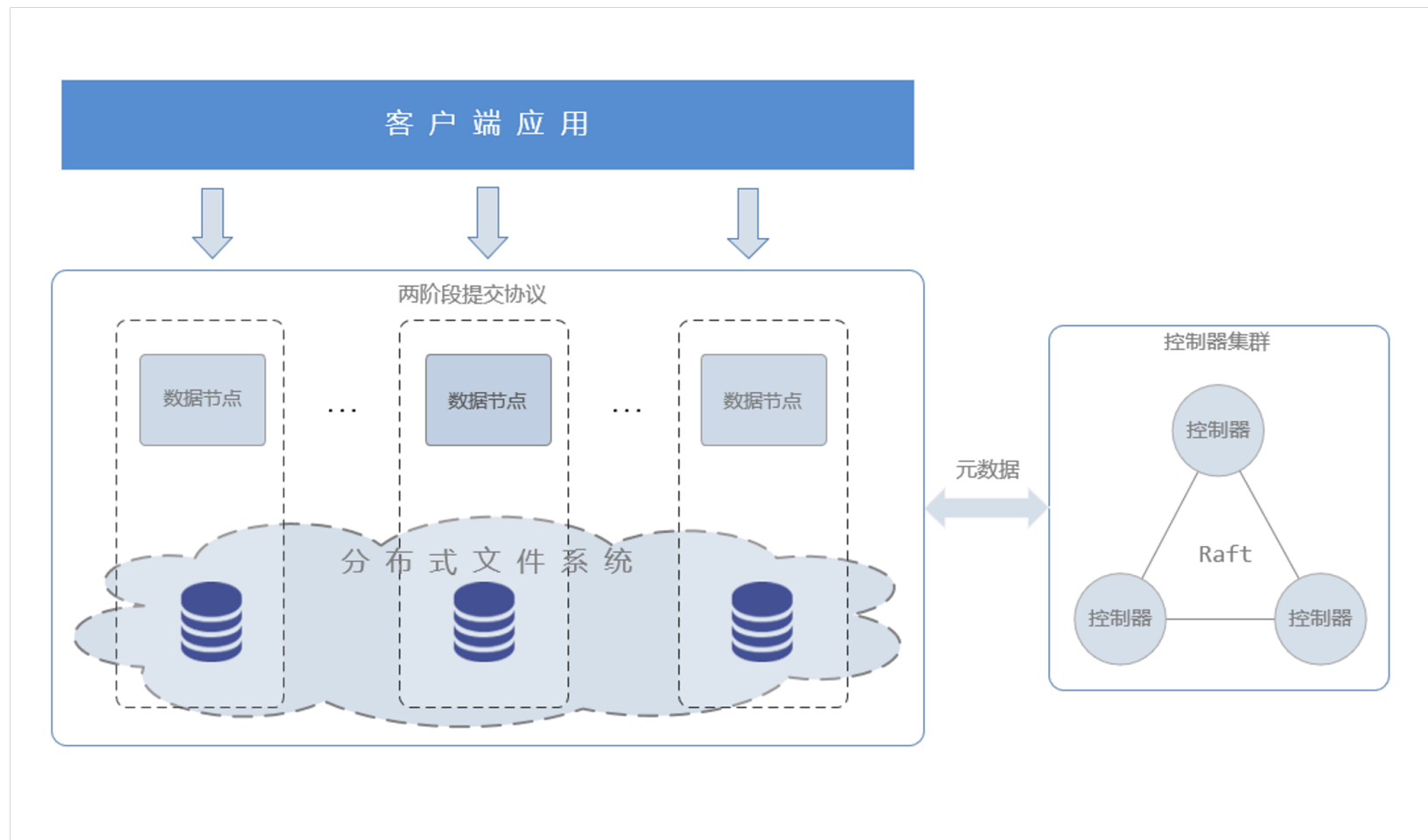


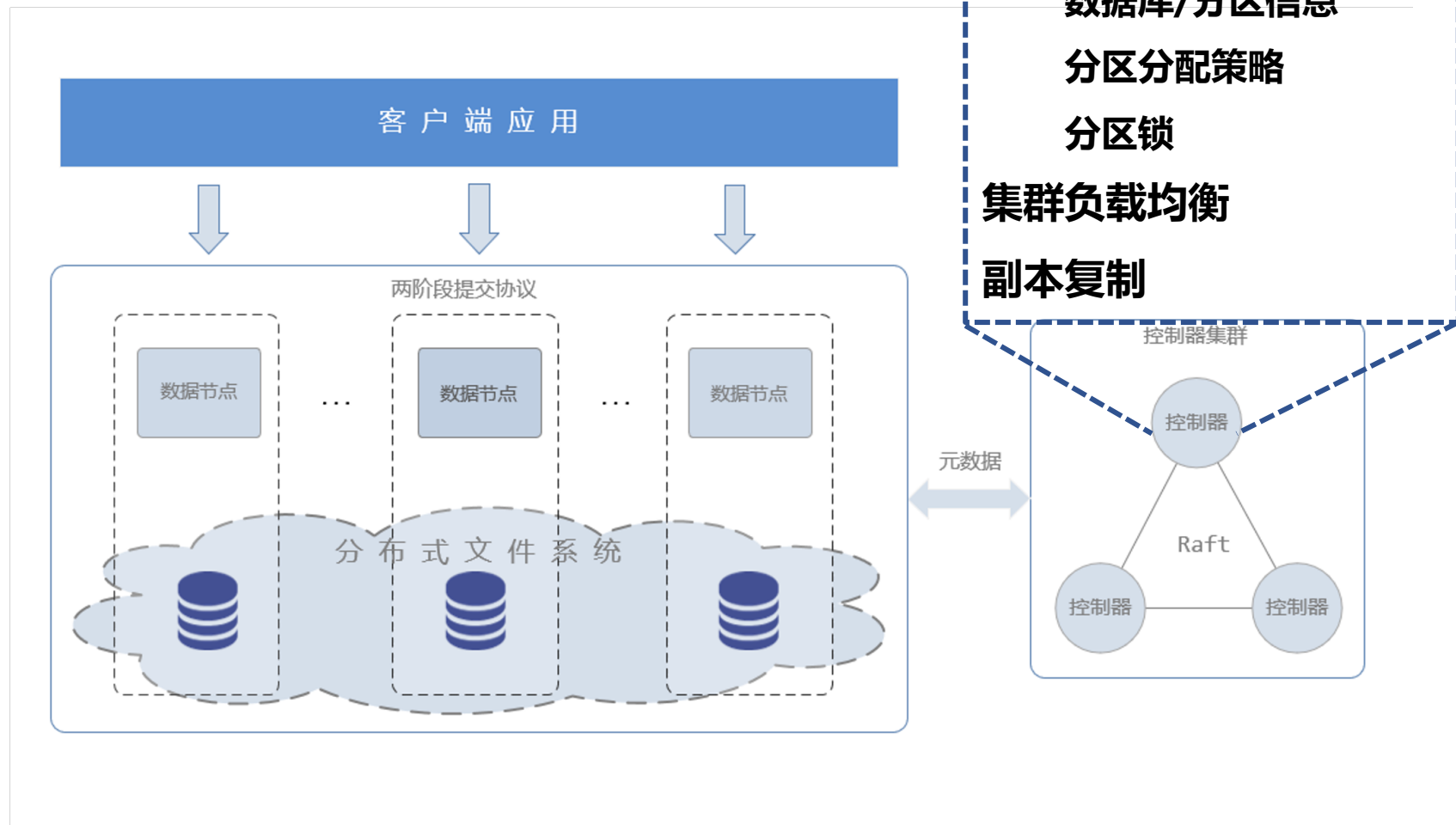


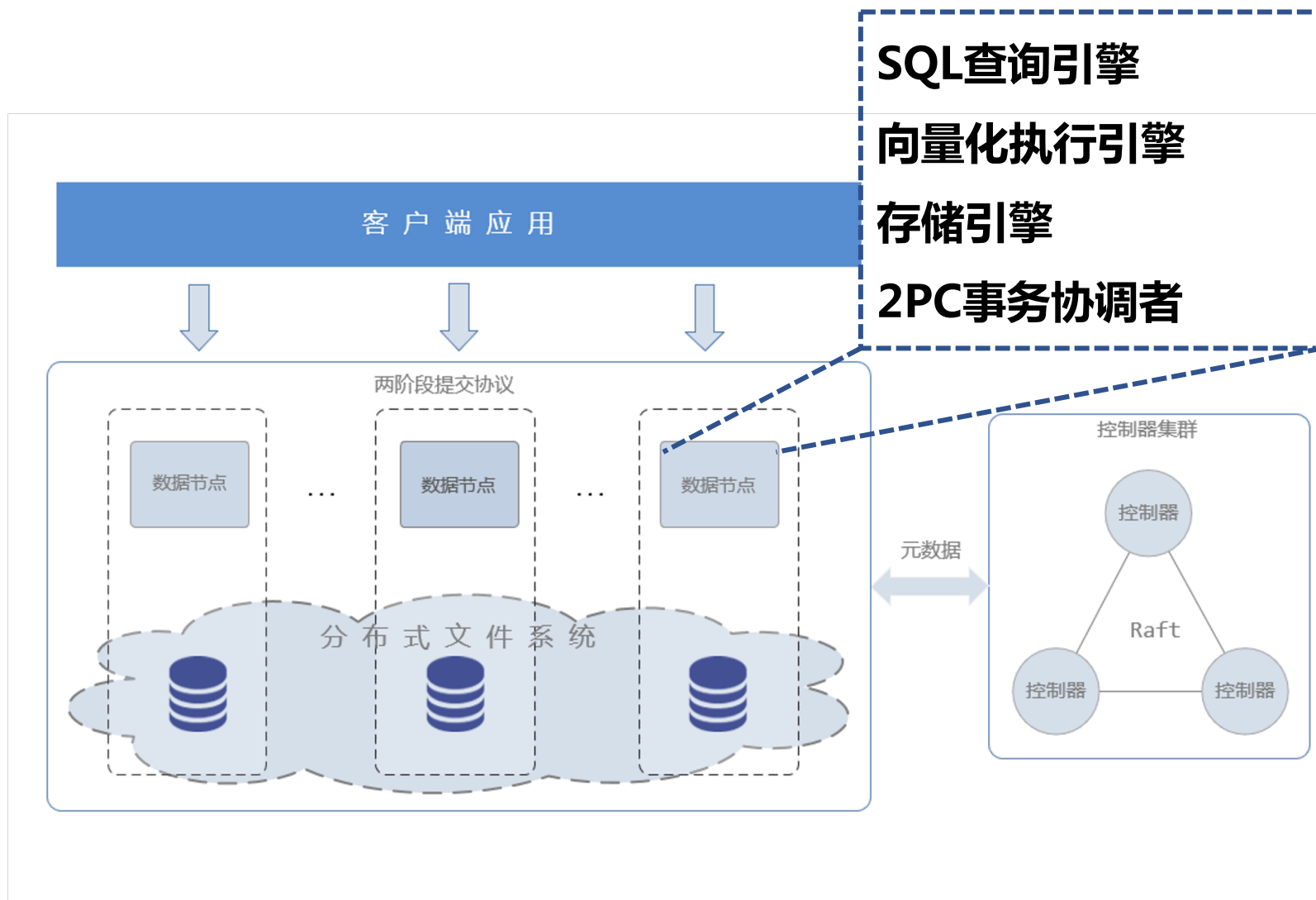


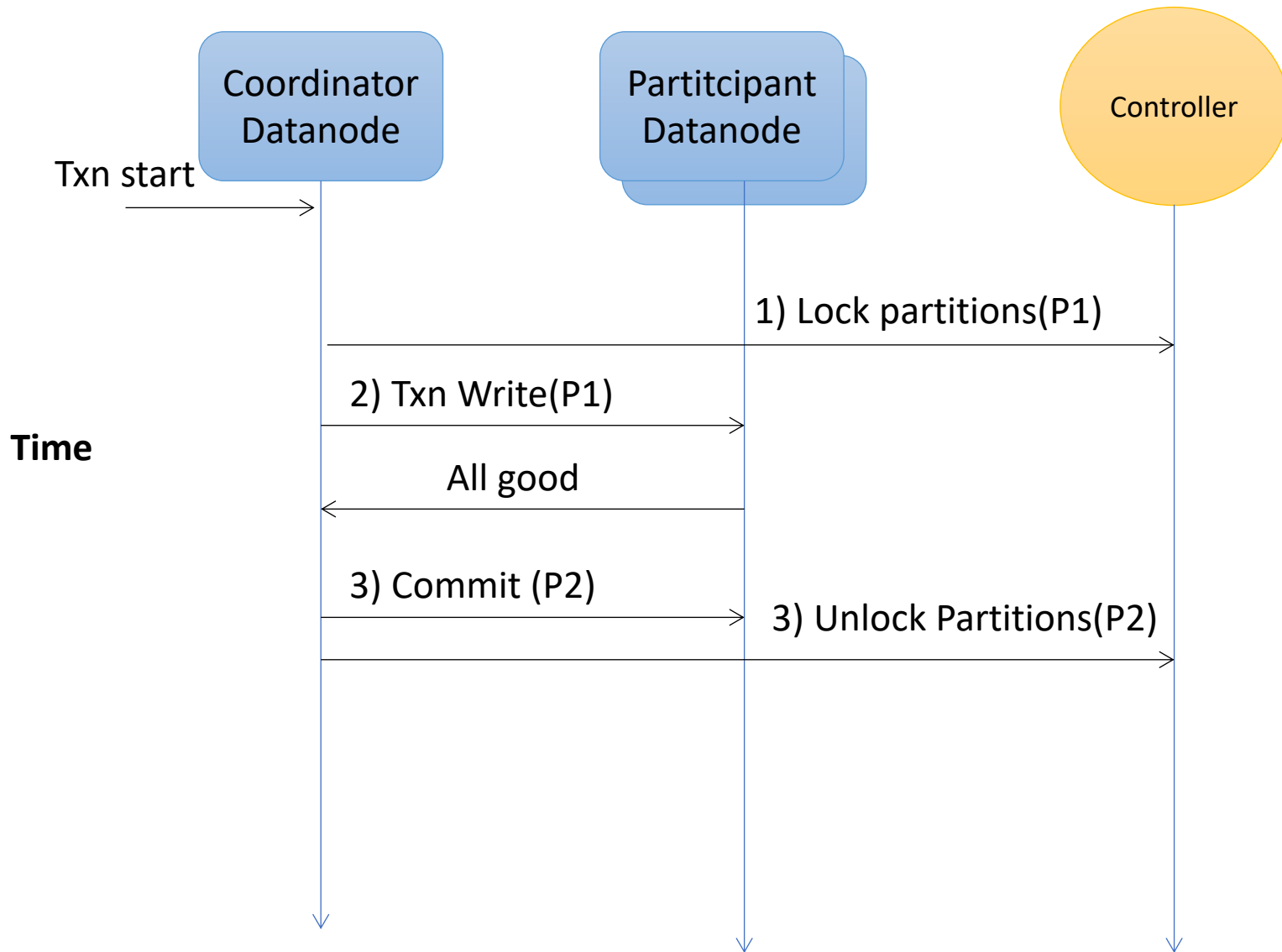


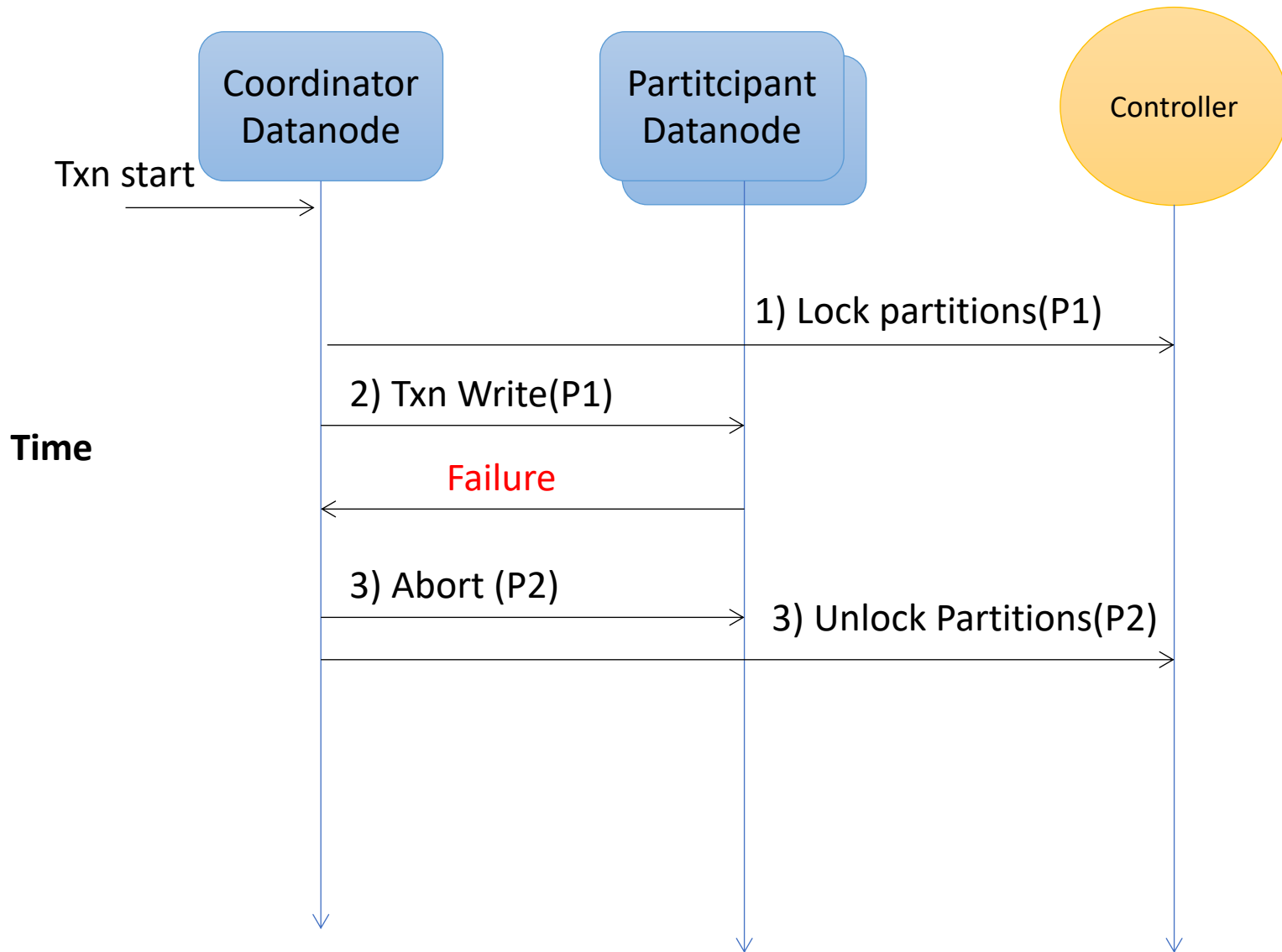
- 时序负载期望写入吞吐高
- 两阶段提交协议(2-phase-commit)
 - 批量写入吞吐高
 - 写入延迟高 - 不敏感





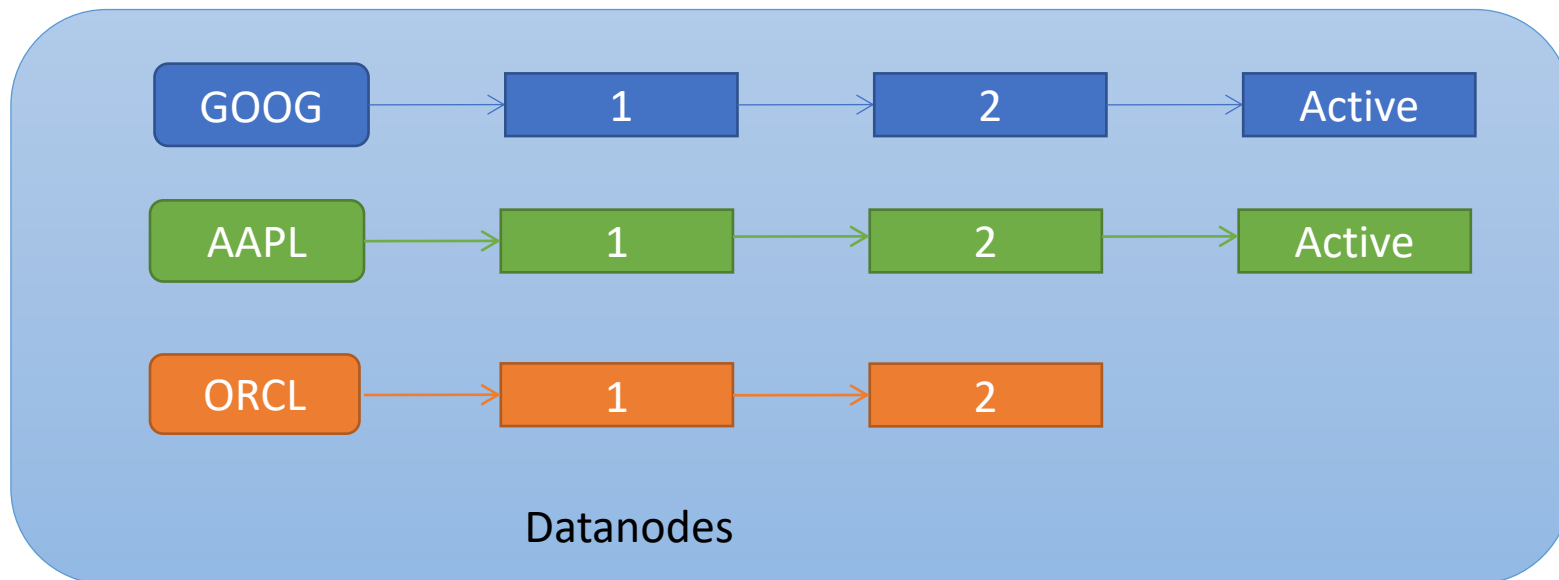


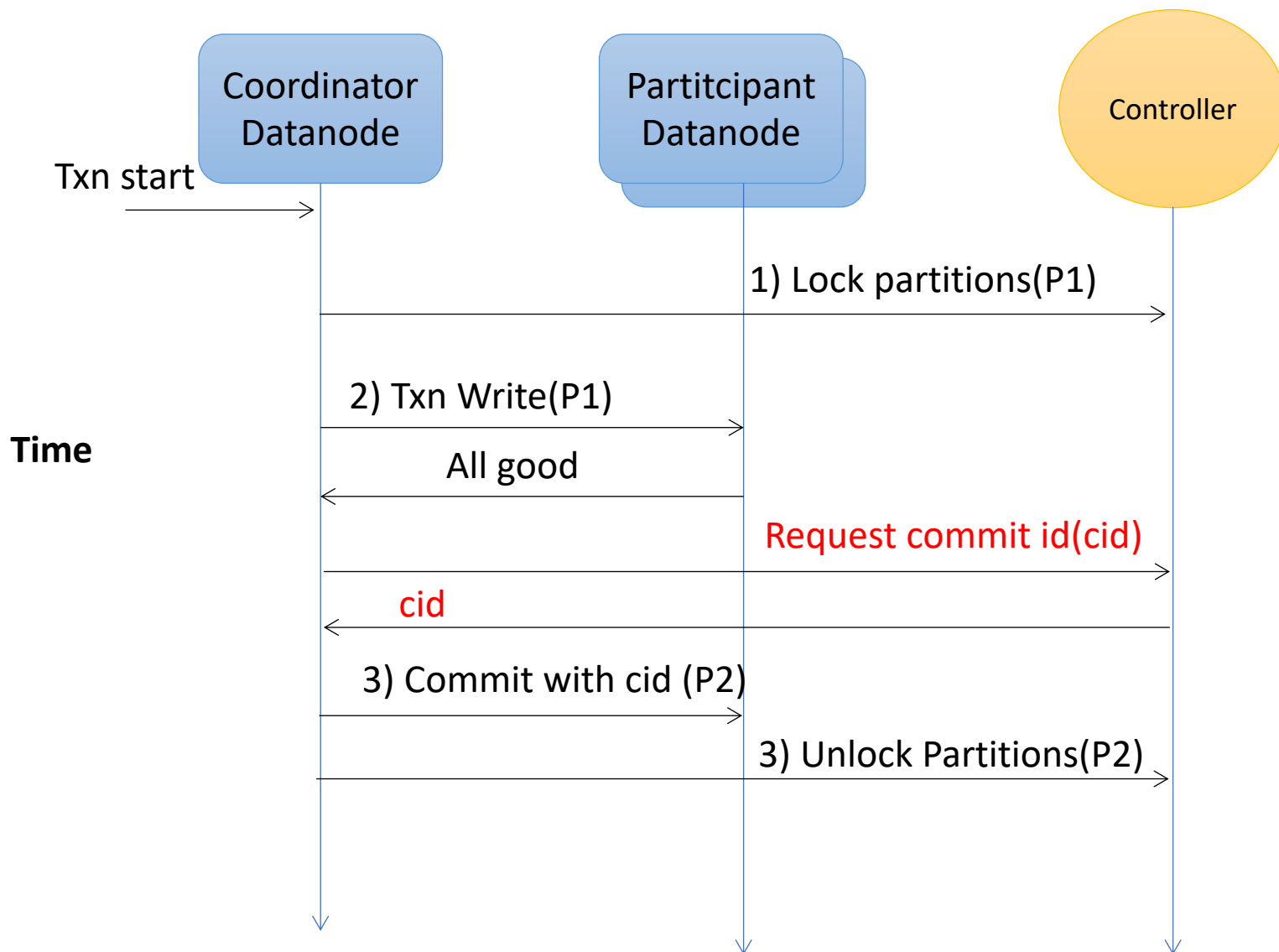




- 写入总是持续不断
 - 千万级别写入每秒
 - 不希望被中断
- 同时需要提供低延迟时序分析查询

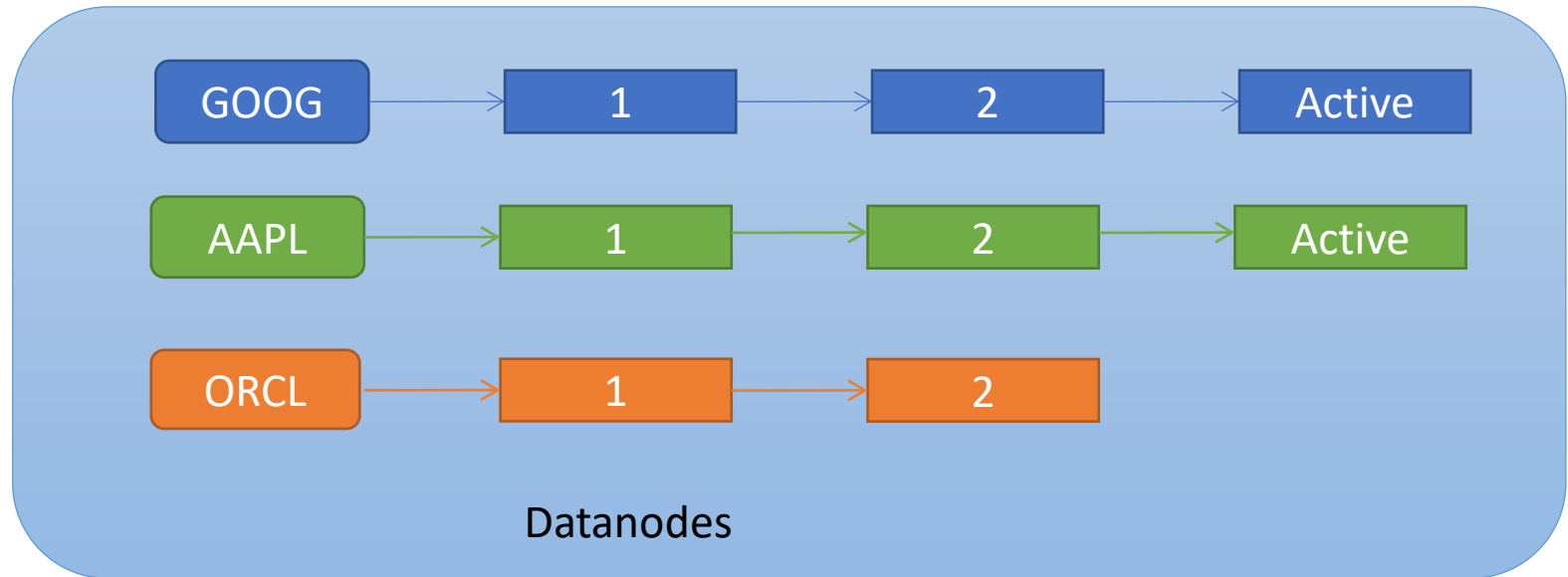
- MVCC - 多版本并发控制
 - Controller确定写事务提交顺序，分配commit id(cid)
 - 分区维护数据版本链，每个事务写入数据关联一个commit id
 - 读写不冲突





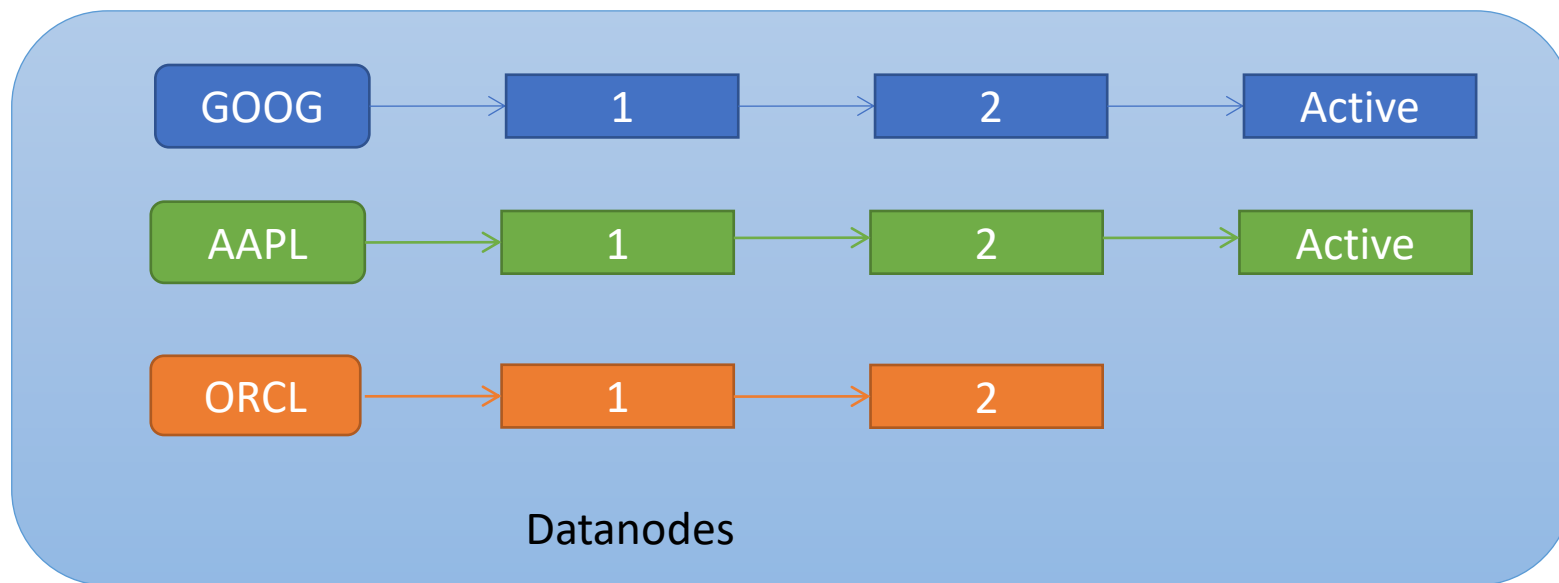
- 存储引擎附加一列cid用以支持MVCC

series sort key		Physical Data Model		Data Columns	Txn commit id
site:group:machine		temperature	pressure	ts	cid
1:1:1		50	101.32	17:01:05	1
1:1:1		51	100.33	17:01:06	2
1:2:3		51	101.32	17:01:05	1



Committed txn cid list : 1 2 3 5

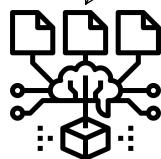
Controller

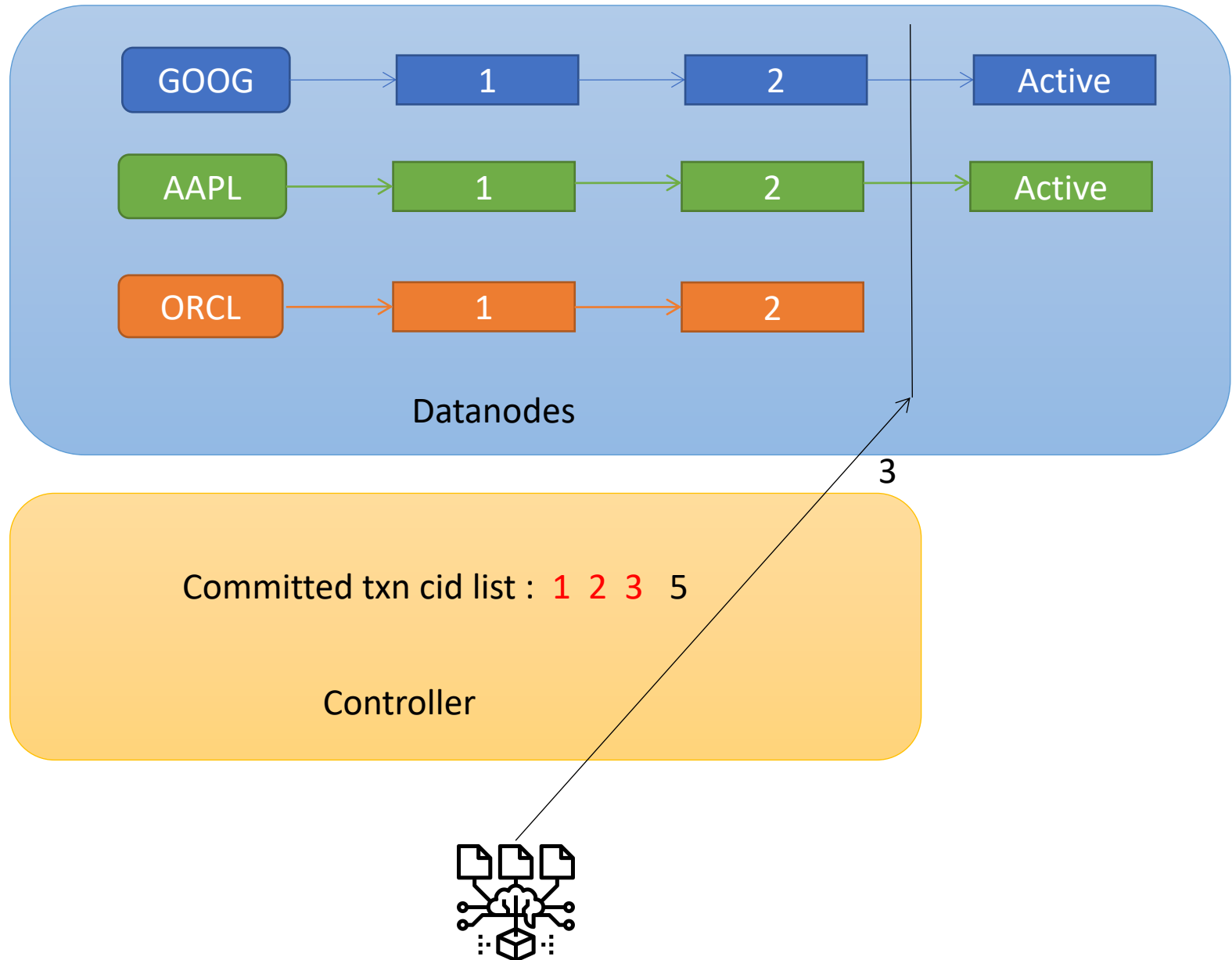


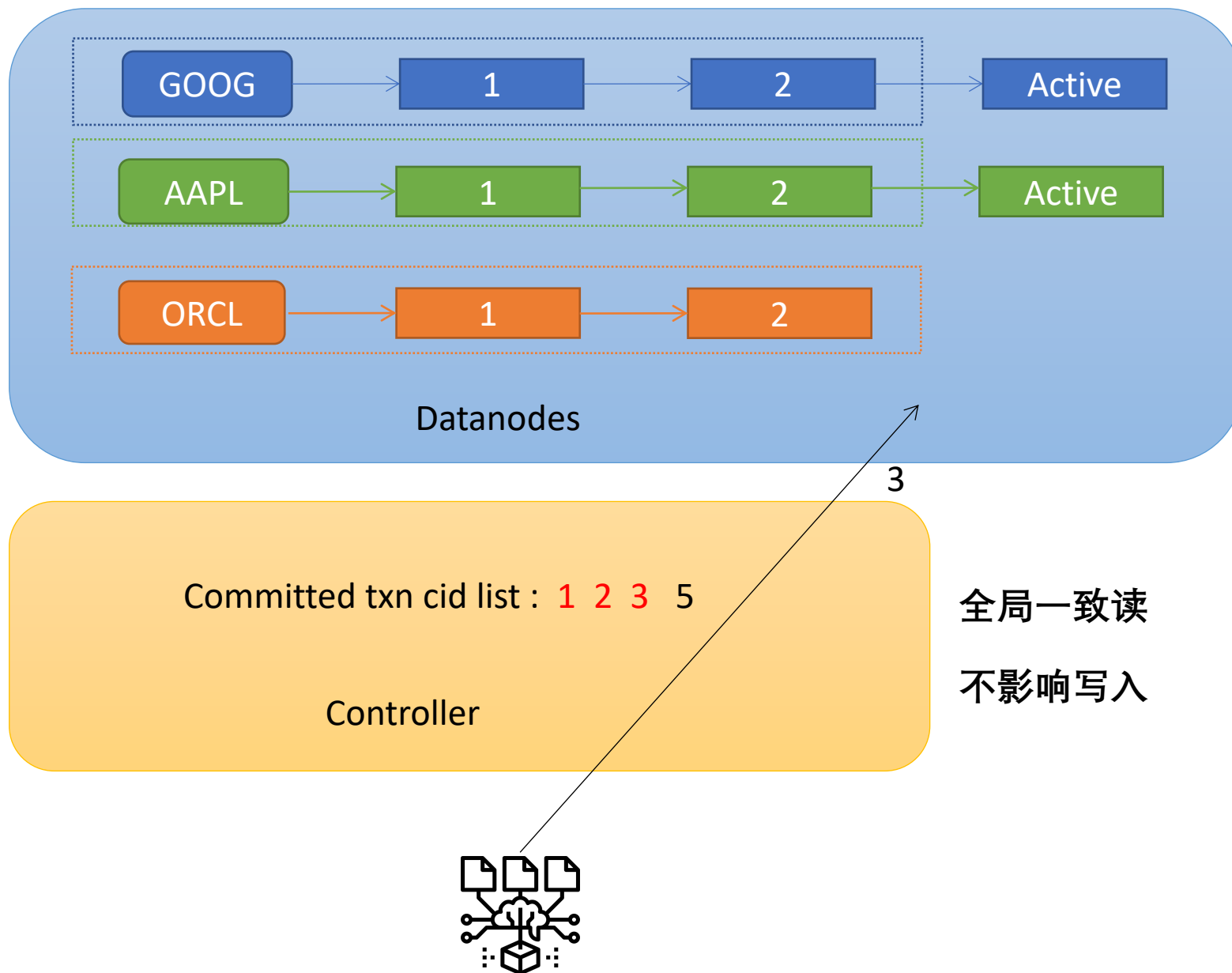
Committed txn cid list : 1 2 3 5

Controller

snapshot cid: 连续已提交的最大事务







- DolphinDB高性能时序数据库
 - 维持高写入吞吐
 - 列式缓存+按时间组织列存数据
 - 减少IO
 - 内存压缩
 - 极速时序数据分析
 - PAX文件格式快速定位时间序列
 - 列存向量化并行化执行引擎+无尽的工程优化
- 事务在时序数据库中的重要性
 - 2PC+MVCC适合时序负载



公众号



技术交流群