GIAC BEIJING Dec.12.16-17

架構 ARCHNOTES

OceanBase数据库架构演进及“双11”实践

杨志丰（竹翁）
蚂蚁金服

技术架构未来

thegiac.com

OceanBase数据库历史

2011.2(v0.1)

2012.4(v0.3)

2014.2(v0.5)

2016.11（v1.2)
支付宝全部
核心链路

2010.6
开幕！

2011.10(v0.2)

2012.11(v0.4)

2015.11
支付宝交易库

网商银行
MYbank

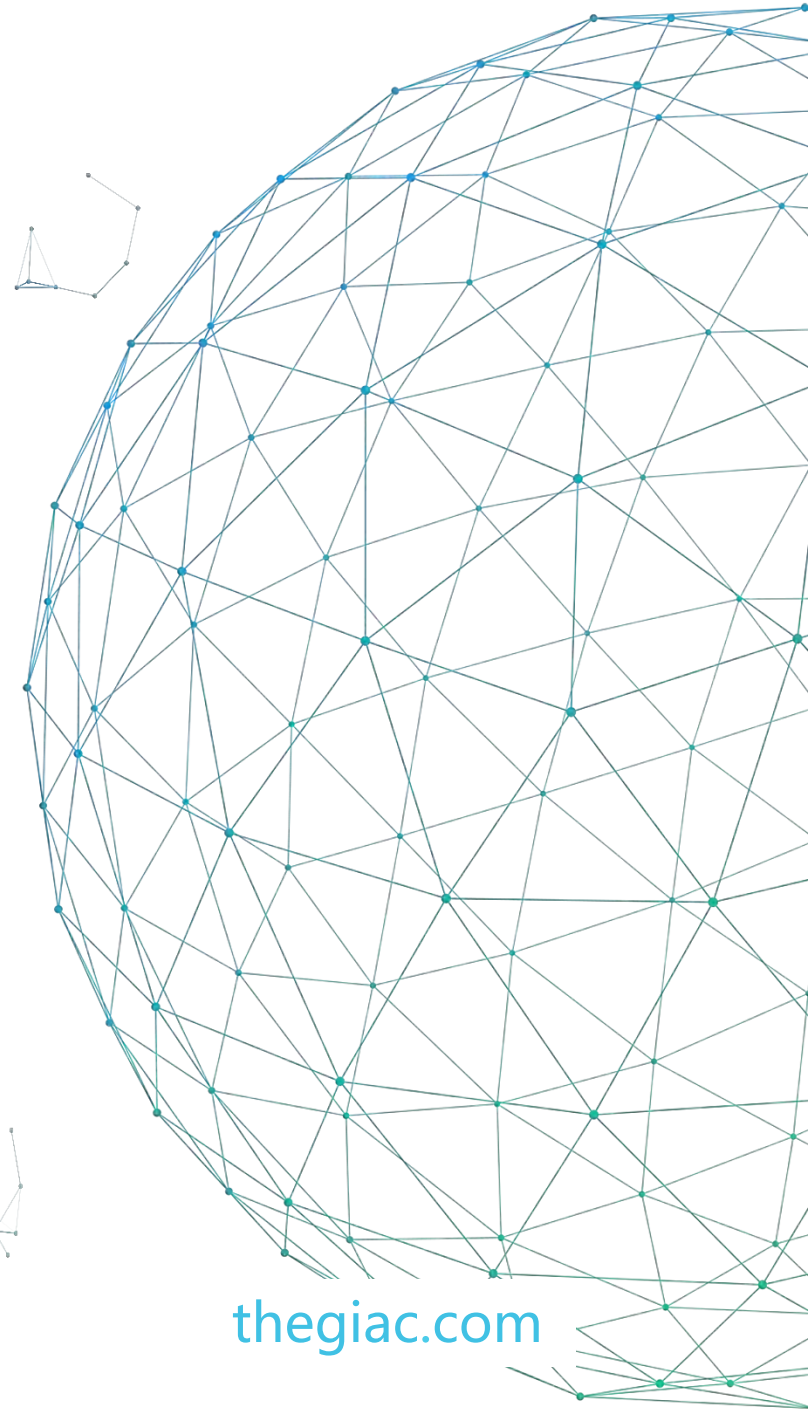
蚂蚁金服
ANT FINANCIAL

GIAC 16-17

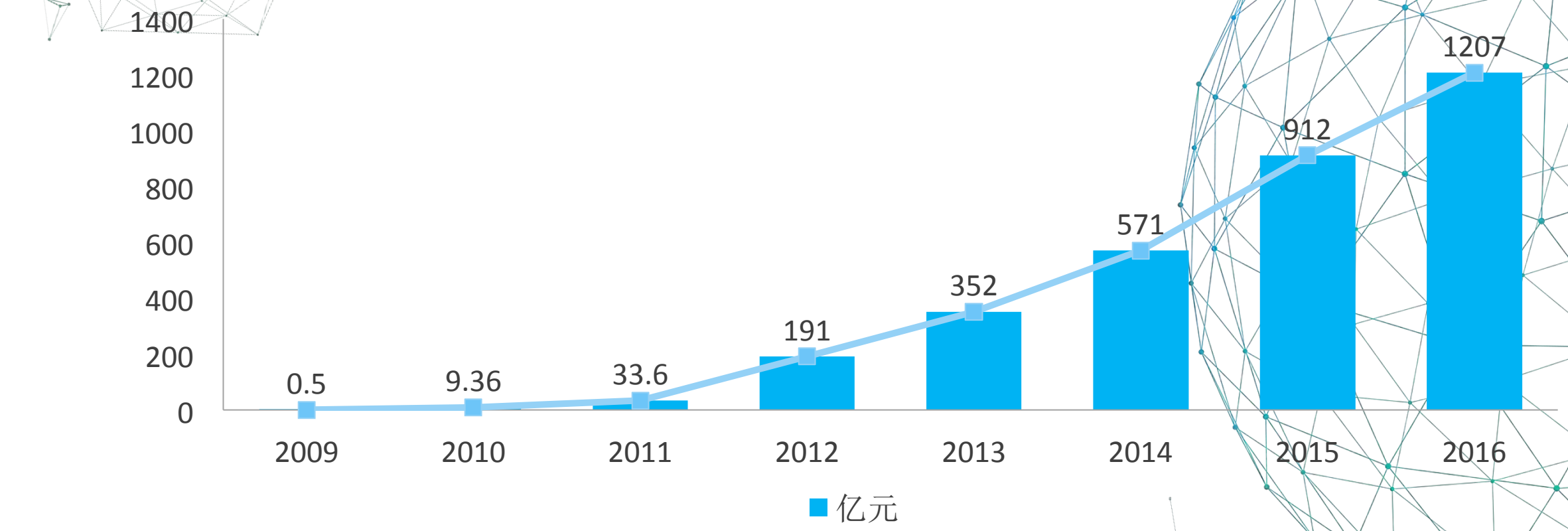
thegiac.com

OceanBase的日常

- 业务
  - 收藏夹
  - 淘宝直通车（OLAP）
  - 支付宝核心：交易、支付、会员、账务、花呗
  - 网商银行：金融云
- 数据
  - 最大单表：200亿行（6400GB）
  - 日常平均QPS：数百万次

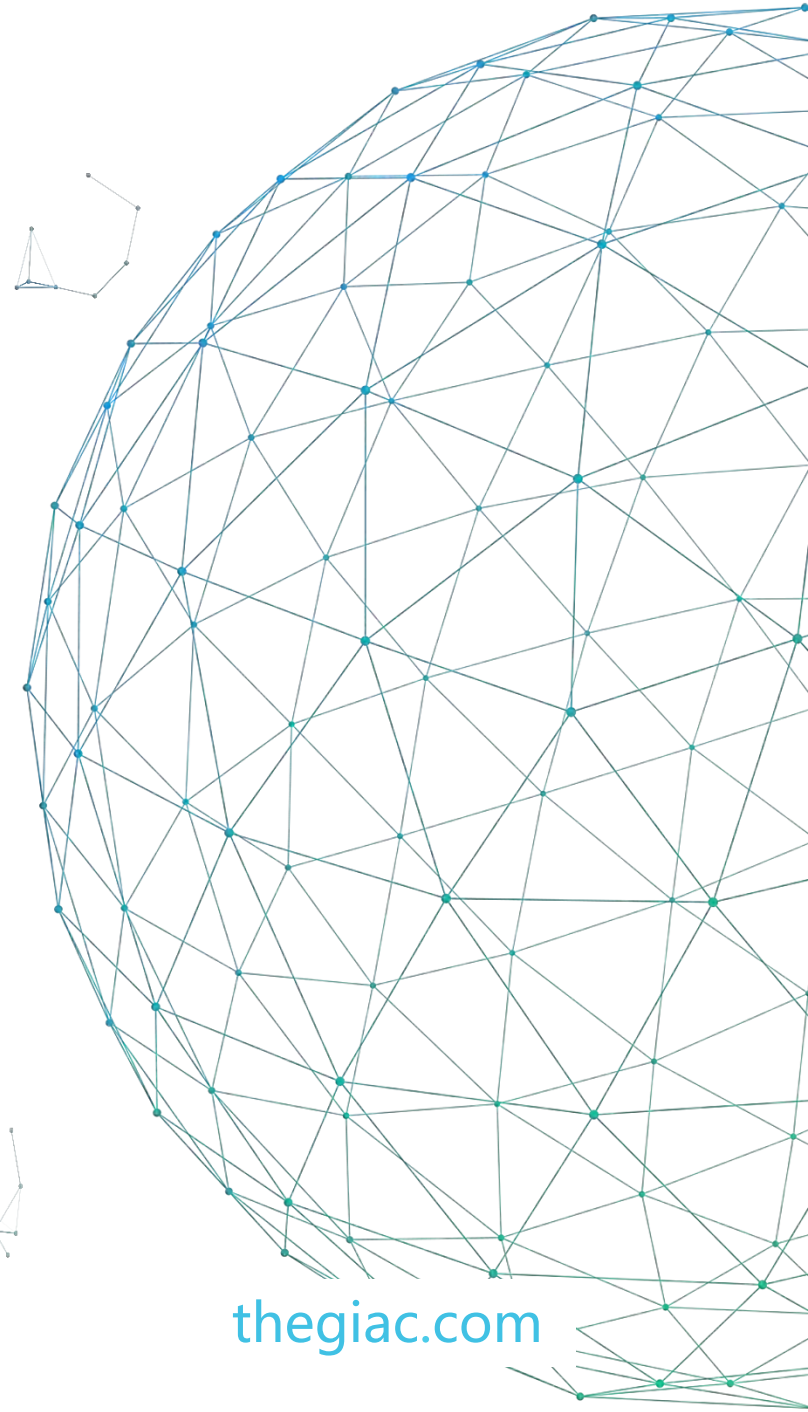


“双11”交易额


1400
1200
1000
800
600
400
200
0
0.5
9.36
33.6
191
352
571
912
1207
2009
2010
2011
2012
2013
2014
2015
2016
亿元

2016年“双11”数据库峰值

- 支付宝交易业务峰值
  - 每秒交易创建：17.5万笔
  - 每秒支付：12万笔
- OceanBase交易库
  - QPS：650万
  - TPS：360万
  - RT：0.7ms

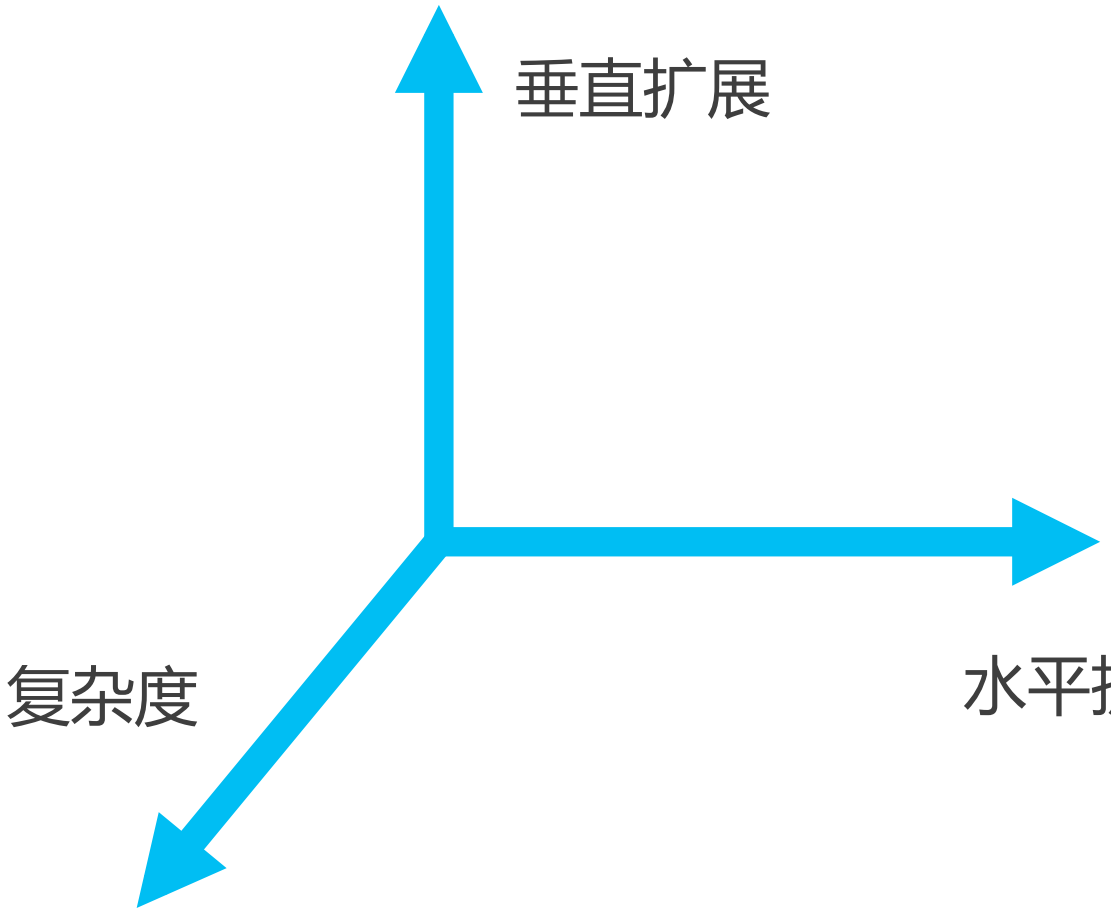


可扩展性

“Scalability is the **capability** of a system, network, or process to handle a growing amount of work, or its **potential** to be enlarged in order to accommodate that growth. For example, it can refer to the capability of a system to increase its total output under an **increased load** when **resources** (typically hardware) are added. ”

-Wikipedia

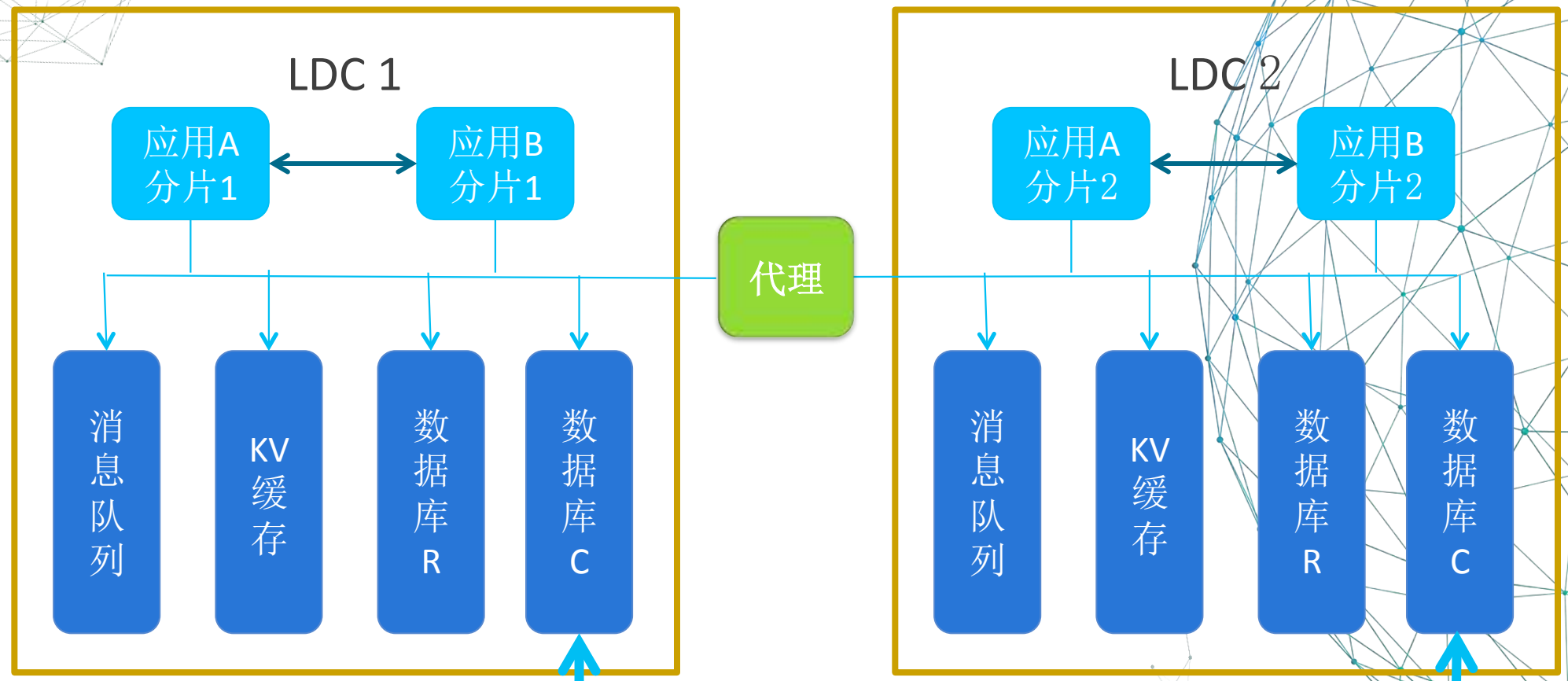
可扩展性



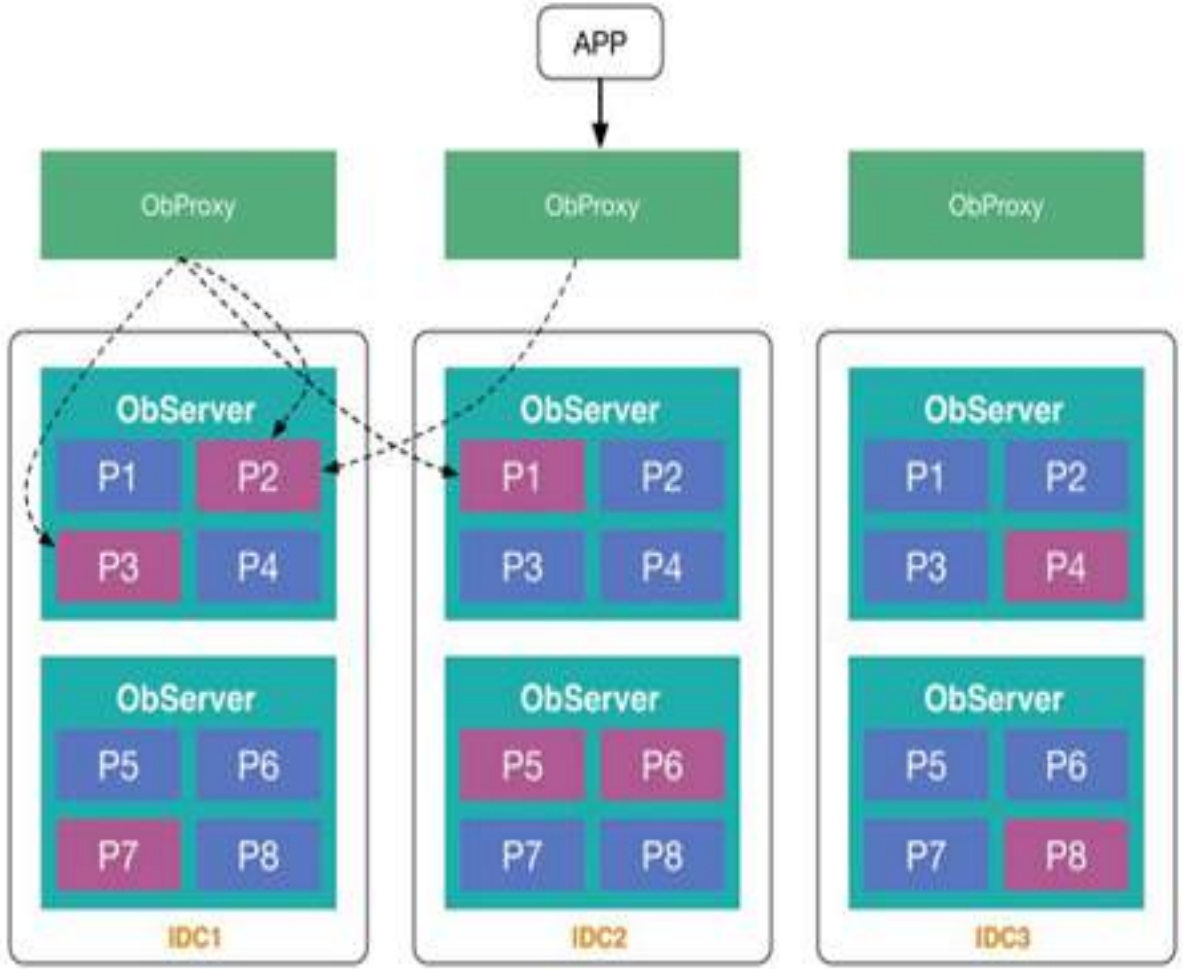
水平可扩展性

- 一个系统通过添加更多的节点而扩容

支付宝核心系统架构

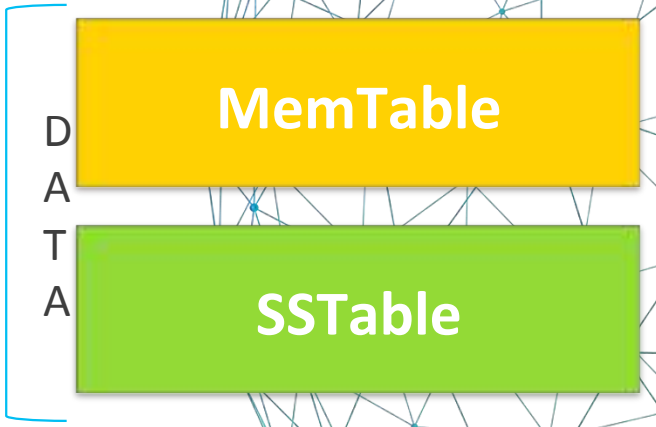

LDC 1
应用A
分片1
应用B
分片1
消息队列
KV缓存
数据库R
数据库C
代理
LDC 2
应用A
分片2
应用B
分片2
消息队列
KV缓存
数据库R
数据库C
复制

OceanBase大图1

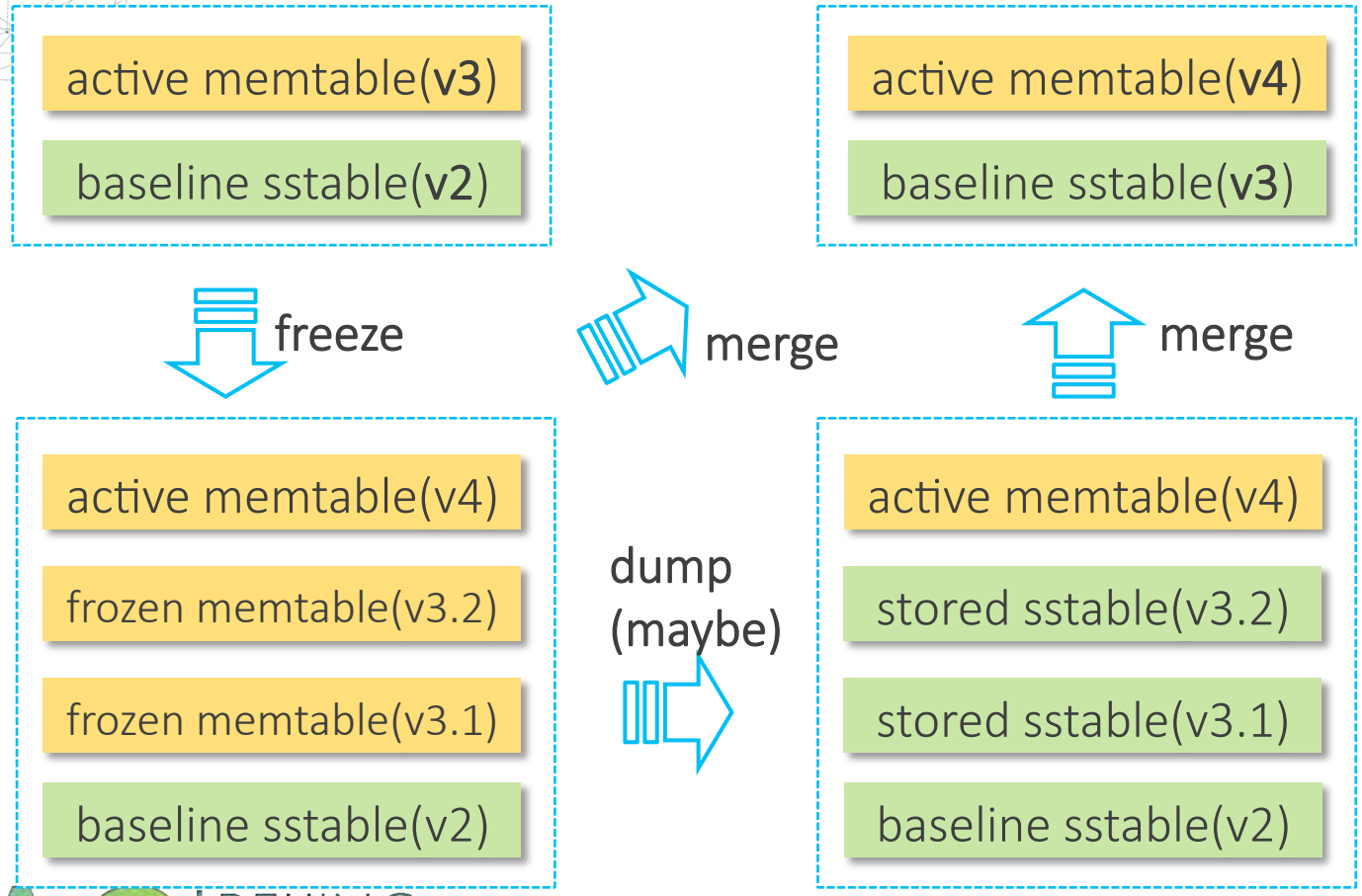


OceanBase的数据模型

- 增删改（增量）数据量较少
  - 假设双11一天6亿笔交易（1200亿/200元）
  - 每个交易1K字节，共600GB
- 两级LSM-Tree
  - 增量数据常驻内存
  - 基线数据只读
    - 变长、压缩、缓存、bloomfilter
- 单份数据的读写分离
  - 无随机写
  - SSD友好

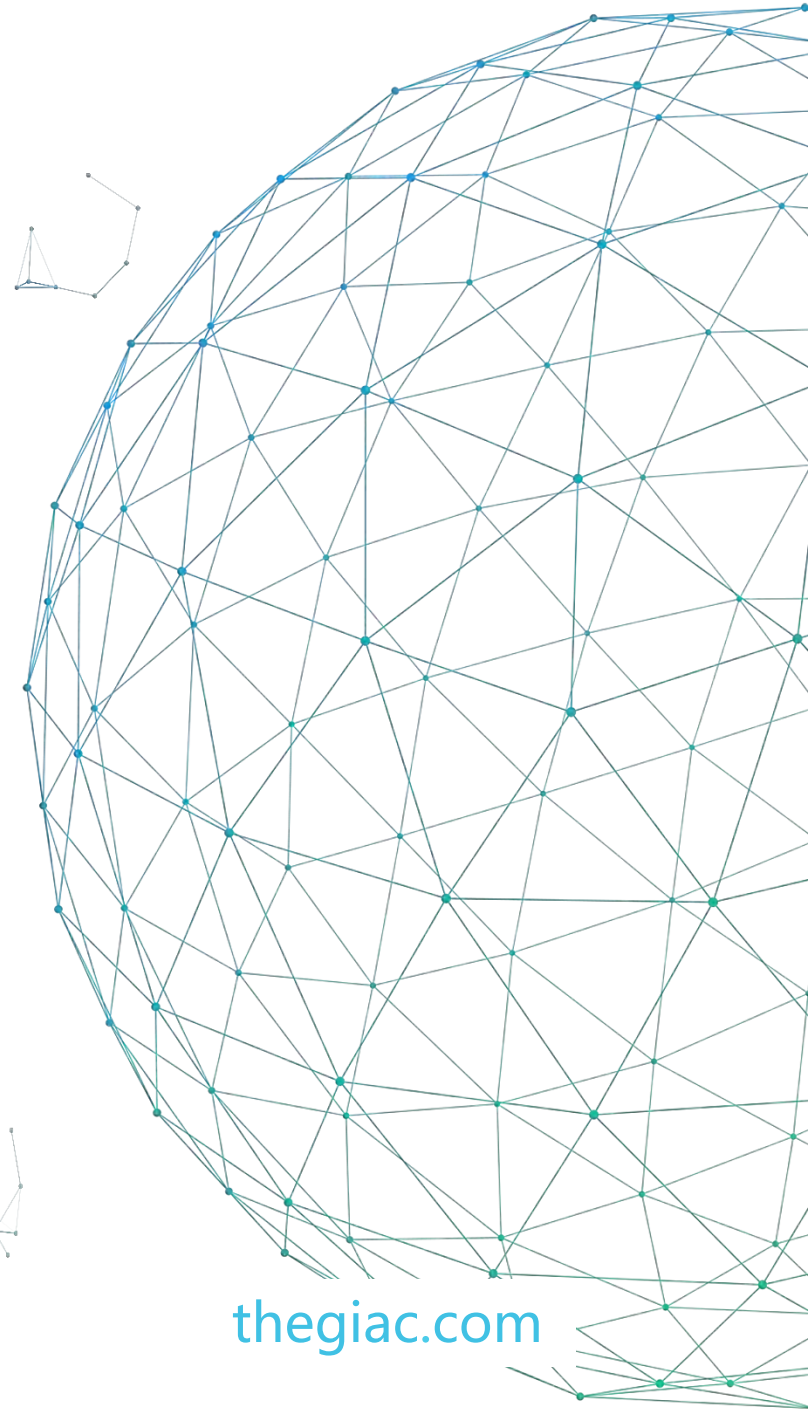


数据模型（cont.）


active memtable(v3)
baseline sstable(v2)
freeze
active memtable(v4)
frozen memtable(v3.2)
frozen memtable(v3.1)
baseline sstable(v2)
merge
dump
(maybe)
active memtable(v4)
baseline sstable(v3)
merge
active memtable(v4)
stored sstable(v3.2)
stored sstable(v3.1)
baseline sstable(v2)

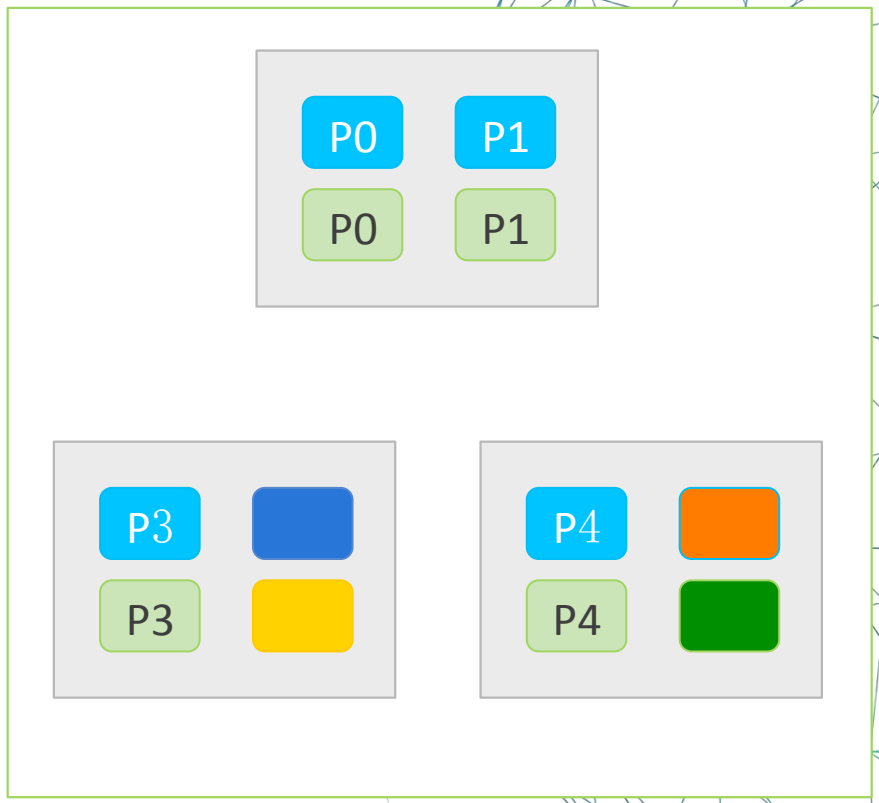
水平分片

- 分表：多表
  - 隔离低；利用率中；SQL支持差；事务支持好
- 分库：多数据库实例
  - 隔离高；利用率低；SQL和事务支持差
- 分区：分区表
  - 隔离低；利用率高；SQL和事务支持好
  - Hash，Range，Interval分区
  - OceanBase 0.5自动分区
  - 二级分区
    - 例如，一级hash uid，二级时间interval



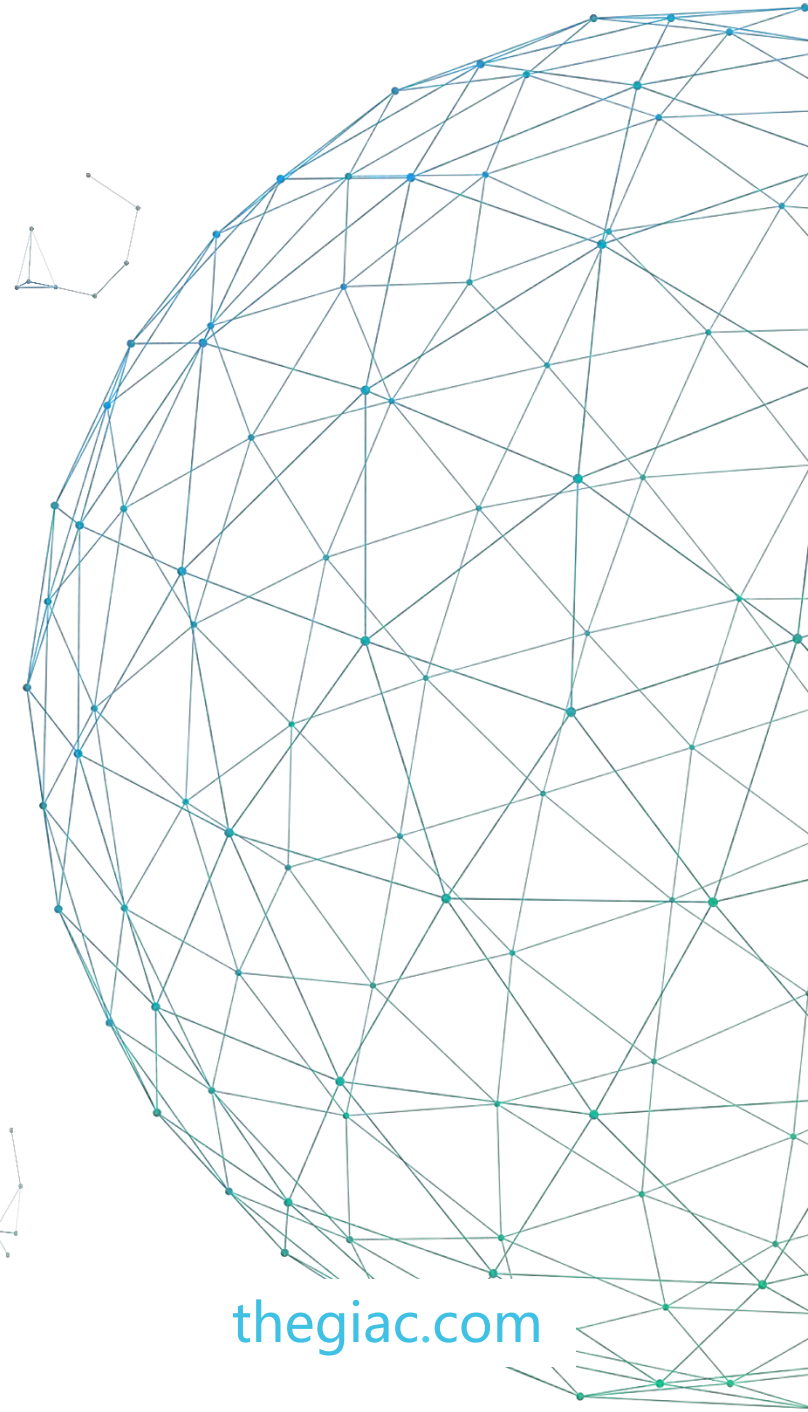
分布式事务

- 单机事务
  - 内存数据库的MVCC
  - REDO日志Multi-Paxos强同步
- 分布式事务
  - 两阶段提交（2PC）
  - 通过分区调度避免分布式事务
- 业务层分布式事务
  - XA Transaction
  - 灵活 vs. 复杂 vs. 性能

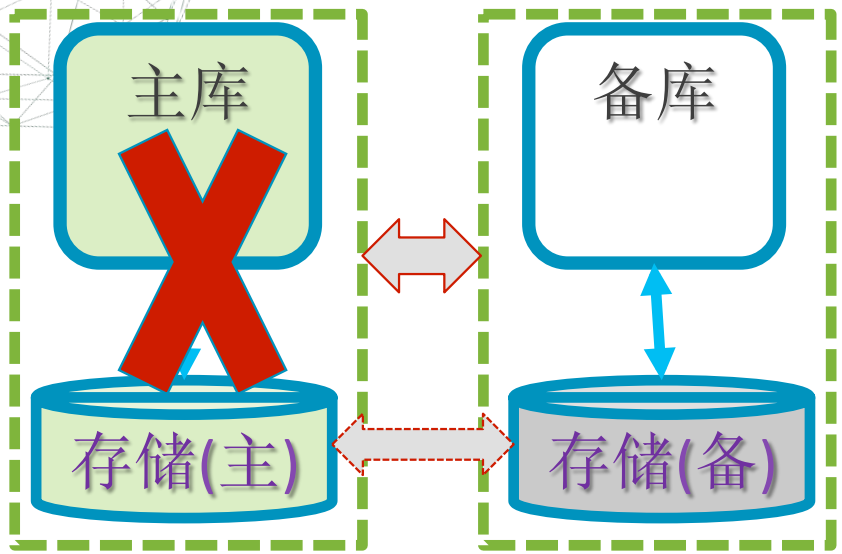


故障&容灾

- 硬件故障
  - 磁盘故障：RAID
  - 单机故障：OceanBase多副本
  - 机房故障：OceanBase同城多中心
  - 城市故障：OceanBase异地多中心
  - 大集群的故障率和系统隔离
- 软件故障
  - 服务多副本
  - 关键算法多份实现（OB 0.5中的例子）
- 人为故障
  - 备份&恢复

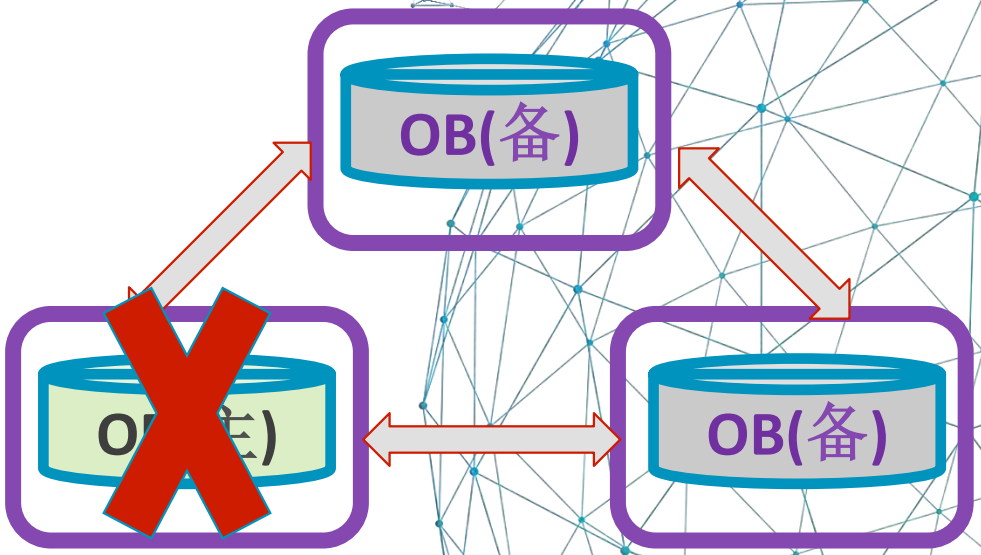


高可用性



传统关系数据库：主备库

☑最大保护模式
☑最大性能模式
☑最大可用模式

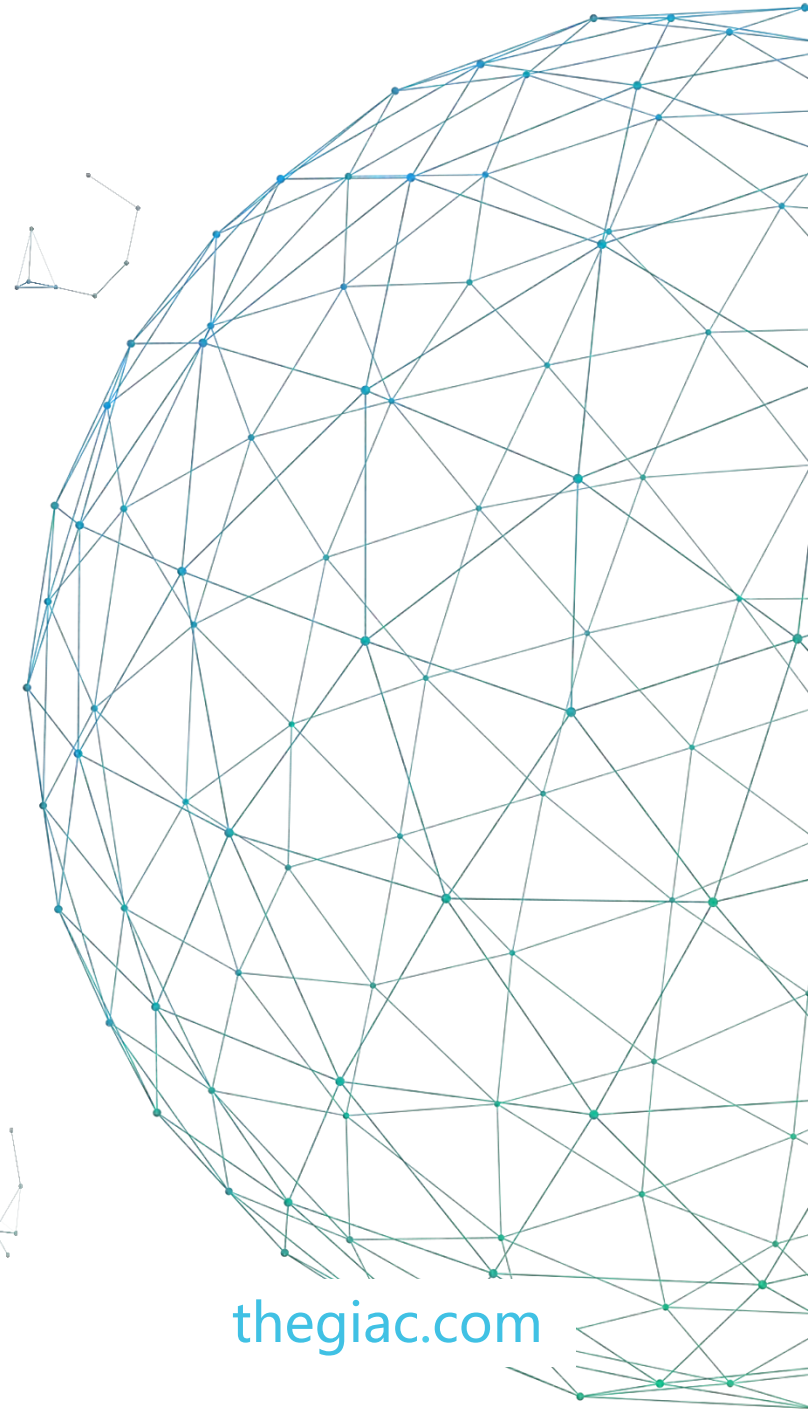


OceanBase：Multi-Paxos一致性协议&分布式选举

☑Paxos协议强同步
☑主备切换不丢事务
☑故障恢复时间小于20s

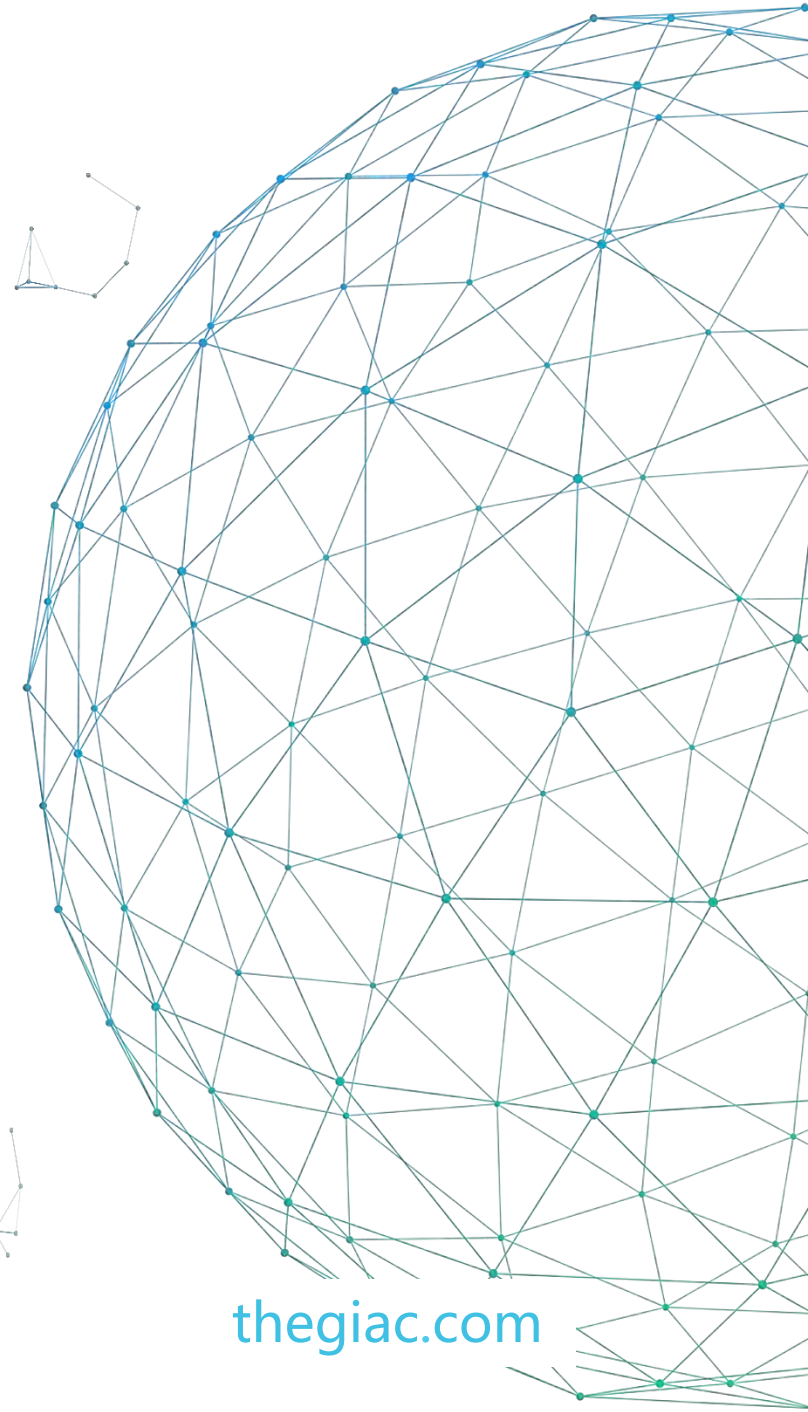
多副本

- 多副本
  - 每份数据有多个副本
  - 每个副本分布在不同的机器上
  - 可以控制不同zone，不同region的副本数
- 副本类型
  - 日志型副本
    - 成本
  - 只读型副本
    - 读写分离
    - 热点数据：读请求水平扩展



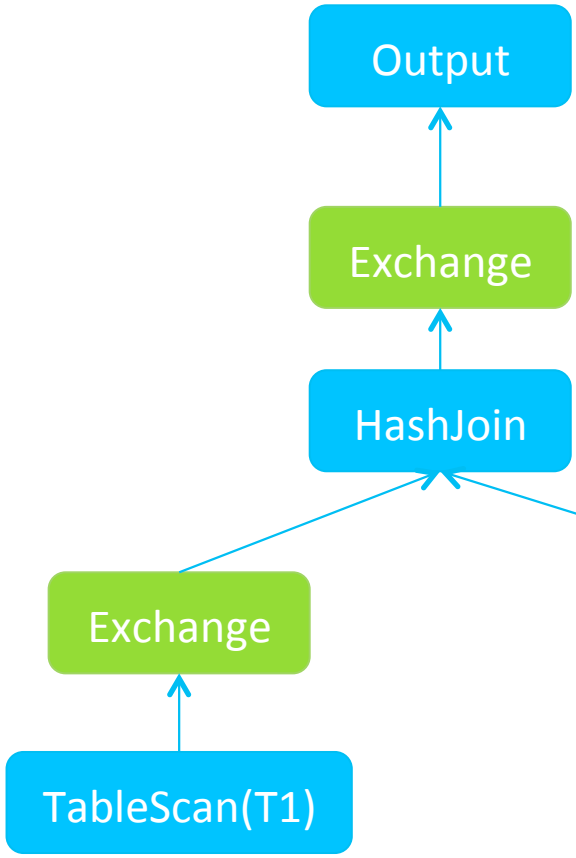
SQL特性

- 完全兼容MySQL的协议和语法
  - 类型系统比MySQL更完善
- 基于代价的优化器
  - 串行计划&并行计划
- 全局计划缓存&自动参数化
- 智能range和分区裁剪
- 大查询并发查询
- 更完善的诊断监控
- DDL变更不锁表
- 函数索引、CTXCAT……



统一的查询引擎

- 执行计划类型
  - 本地执行计划
  - 远程单机执行计划
  - 分布式执行计划
- 优化事务层接口



分布式查询的执行

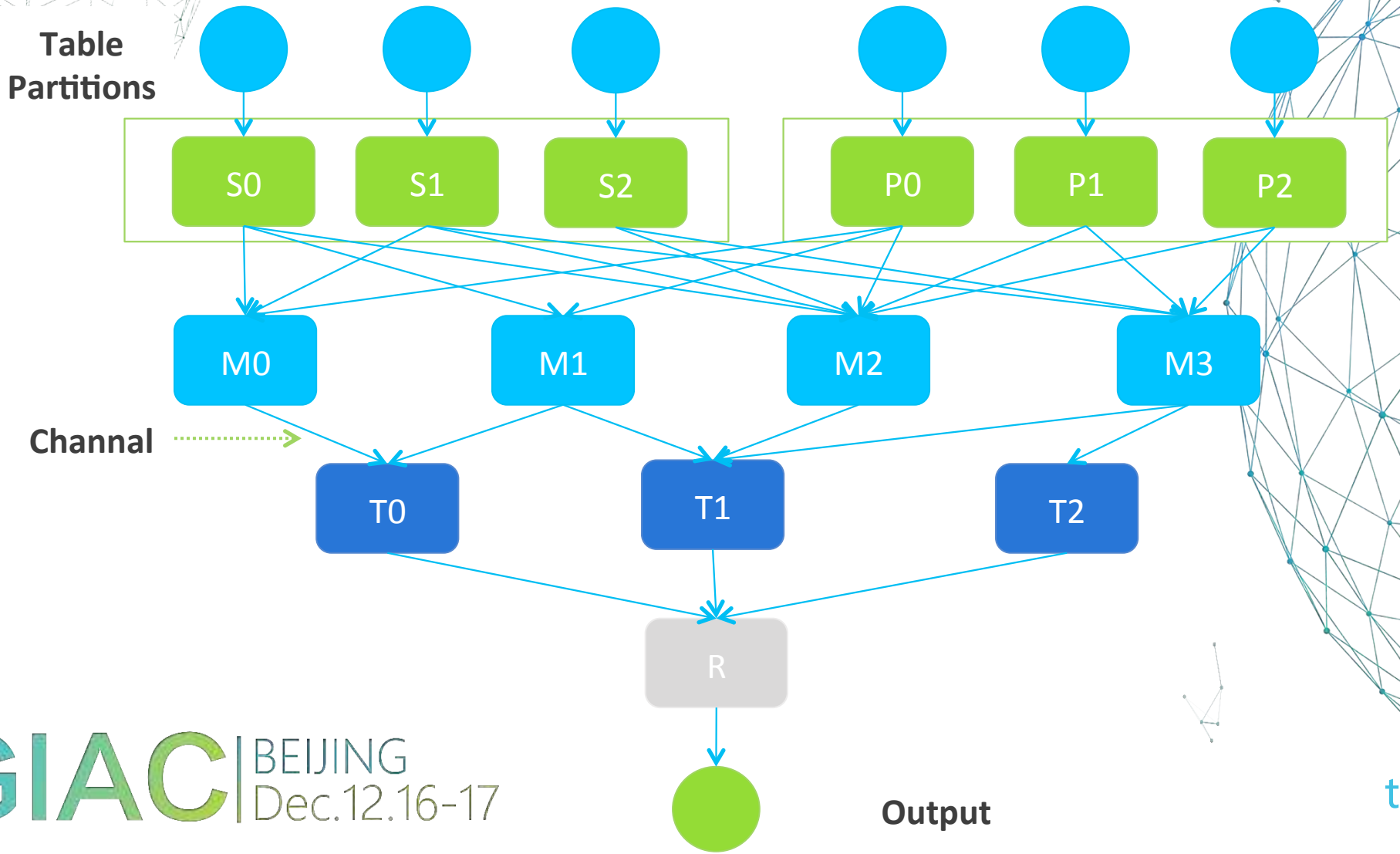
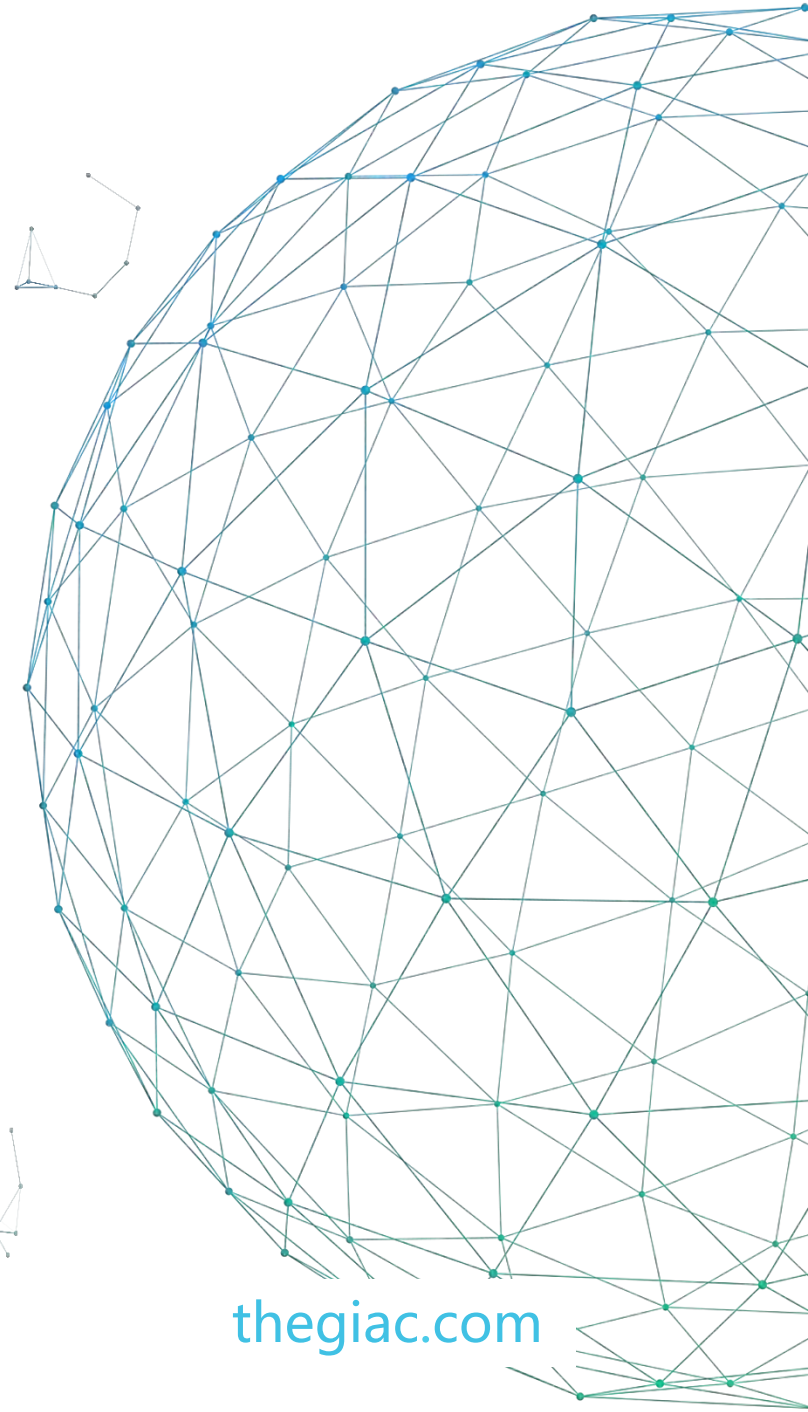


Table
Partitions
S0
S1
S2
P0
P1
P2
M0
M1
M2
M3
Channal
T0
T1
T2
R
Output

客户端库和Proxy

- 查询路由及自动容灾
- Proxy vs. 客户端库
  - 客户端库只提供JDBC Driver
  - 数据库连接（session）保持
- Proxy
  - 部署方式
    - 客户机本地部署
    - 集中部署
  - 热升级

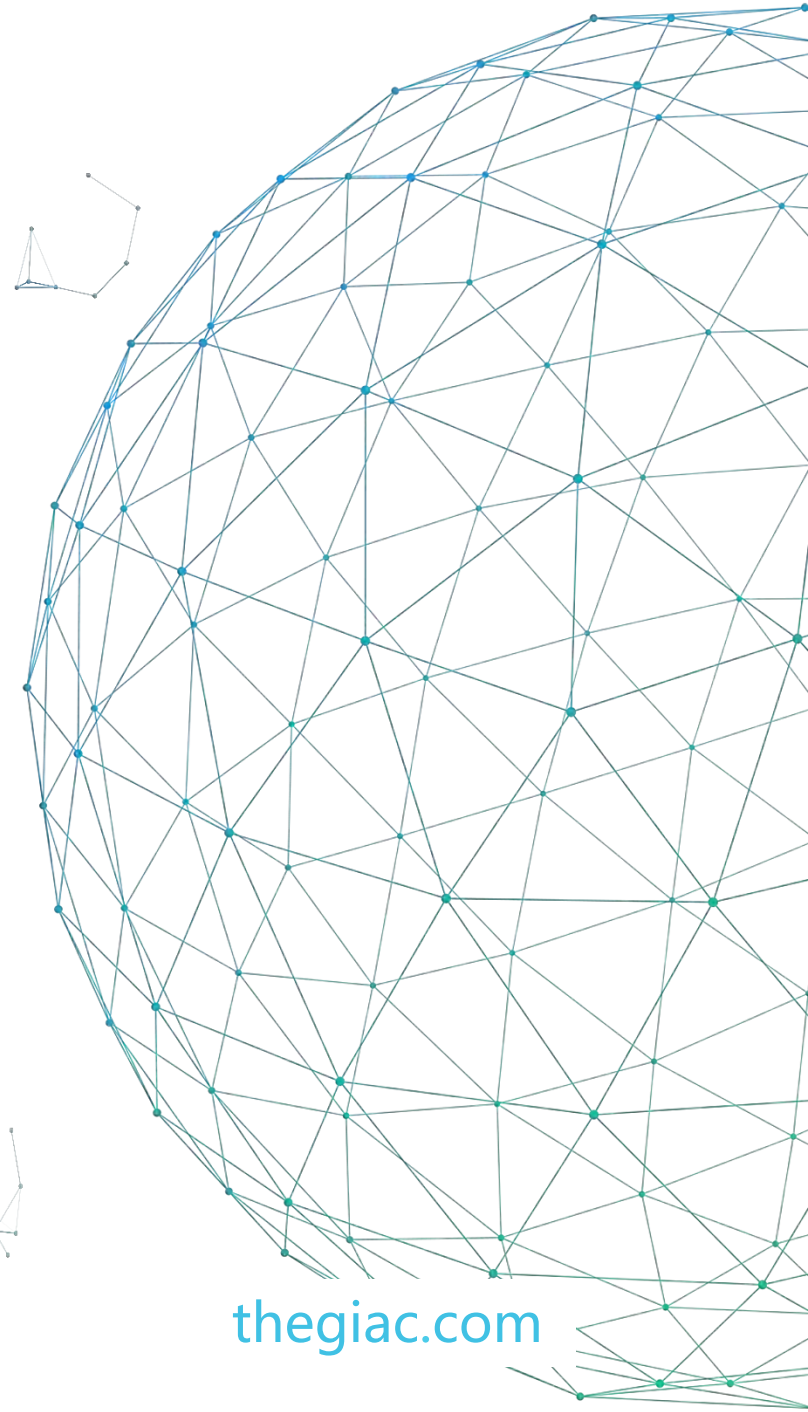


垂直扩展性

- 在一个节点中添加更多的资源（CPU、内存）而扩容

垂直可扩展性

- 提升单机性能
  - 减少指令
    - 行缓存&hash索引
    - SQL全局Plan Cache和自动参数化
    - 编译执行（进行中）
  - 单机可扩展性
    - 锁，锁，锁
    - 定制内存分配器：内存碎片，生命周期管理
  - 单机并发执行
    - 如利用IO并发get
  - 控制：减少不必要的开销
    - 查询短路径



单机性能数据

	INSERT	SELECT	OLTP
TPS/QPS	181,869	751,808	206,437
平均RT	2.75ms	0.66ms	43.59ms
95%分位RT	4.20ms	1.22ms	57.17ms

测试时间：2016年11月

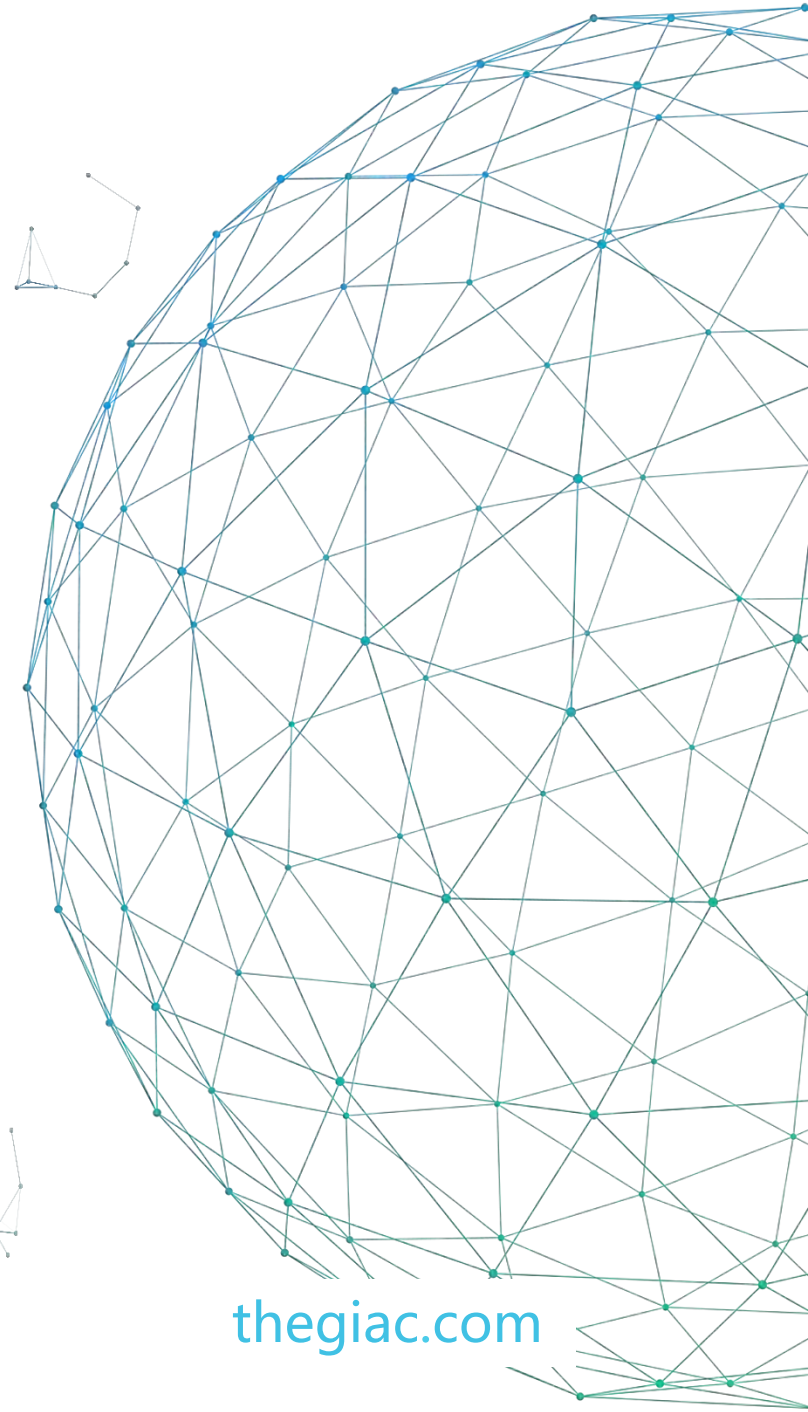
- Sysbench
- 一主两备，仅主库服务
- Intel(R) Xeon(R) E5-2682 v4@2.50GHz×64(HT)

Memory：512GB RAM
Disk：LSI RAID5
连接数：500

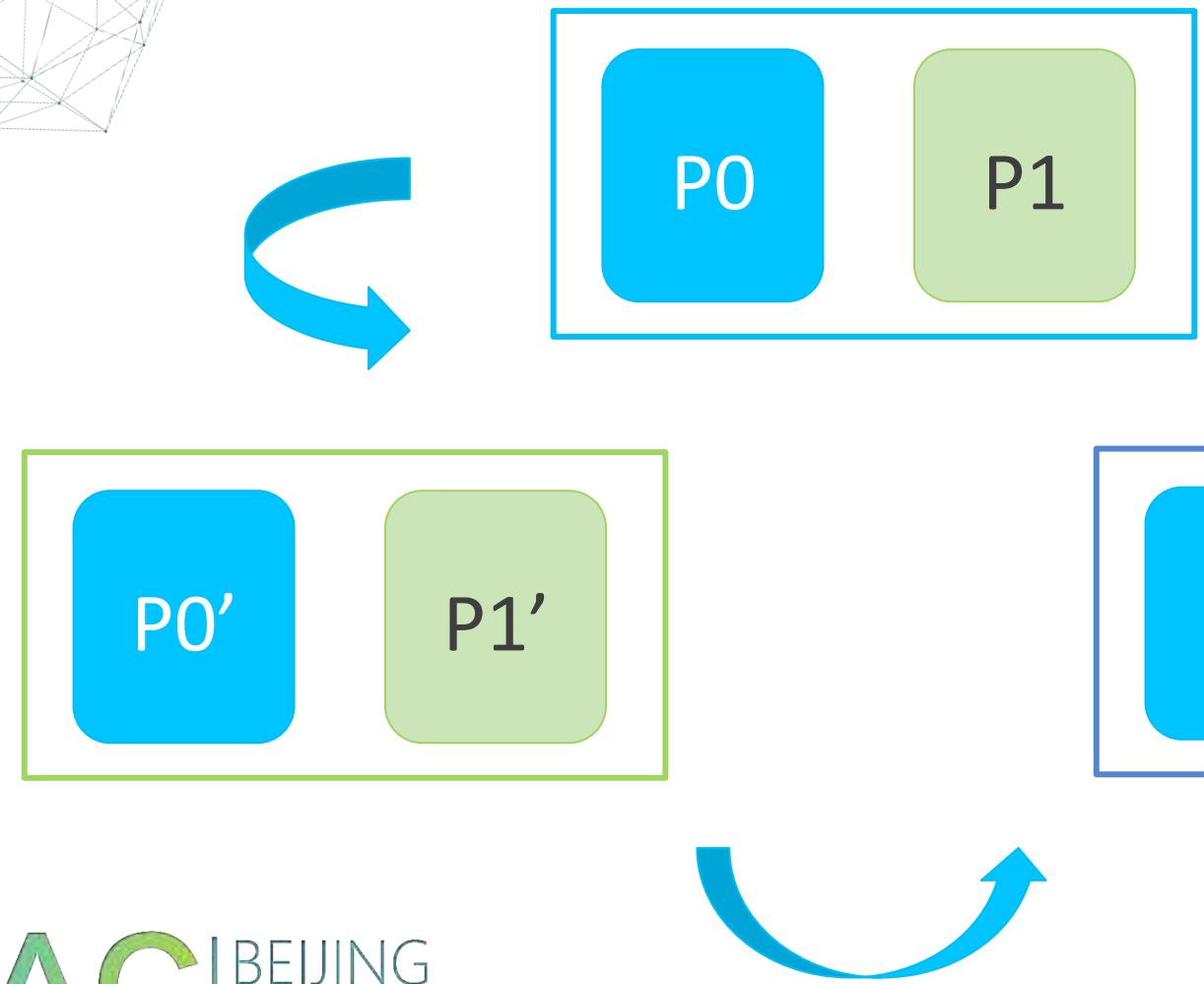
关于读写比的影响
- 1/(0.3 + 0.7/2) = 1.54
- 1/(0.3/2+0.7) = 1.18

垂直可扩展性（cont.）

- 提升单机资源利用率
  - DB内多租户提升整体资源利用率
    - SQL-VM
    - 最大化资源共享
    - CPU共享为例
  - 利用多机特性：批量轮转合并
  - 克服高可用多副本的开销：多库多活

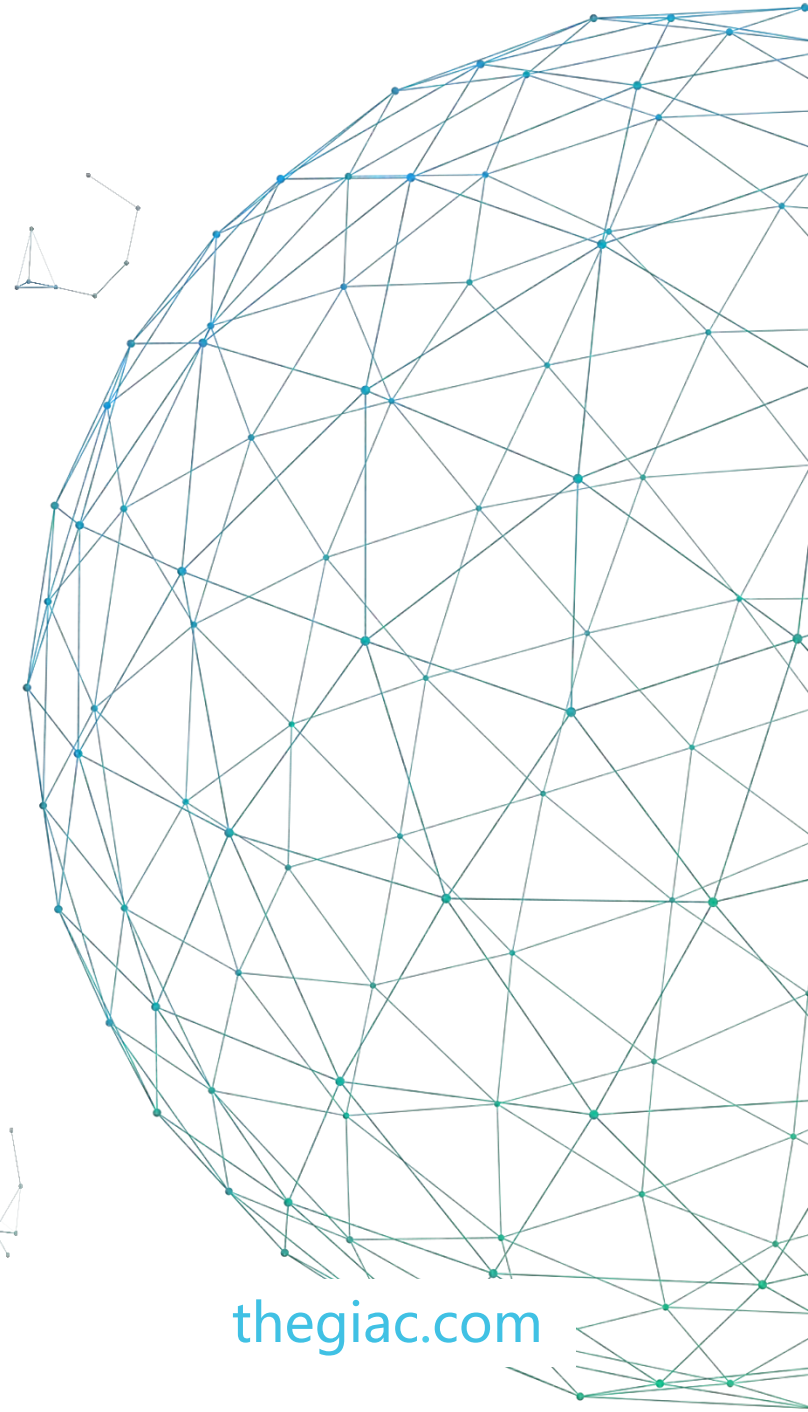


轮转合并



成本小结

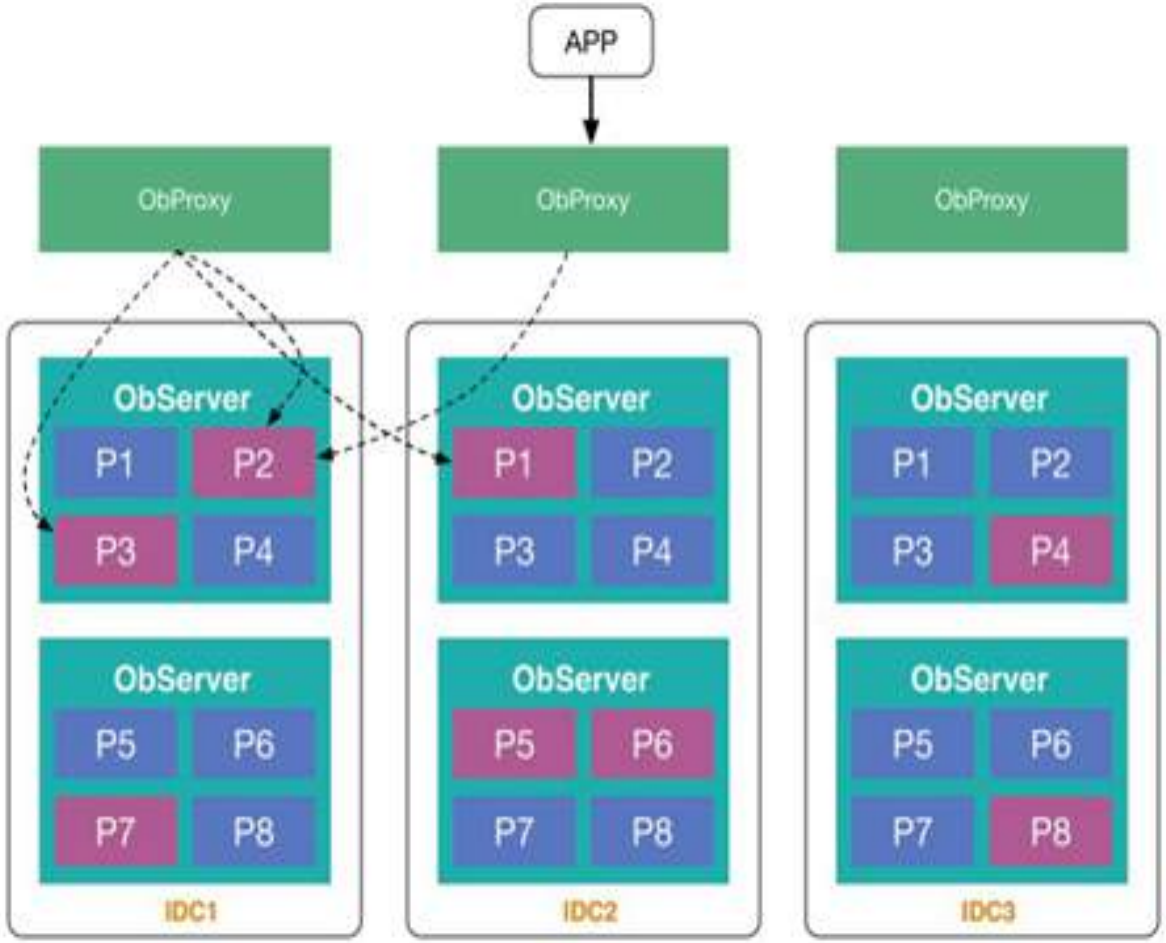
- 和性能的关系
  - 单机性能越高，成本越低
  - 但不只是单机性能
- 提升资源利用率
  - 多库多活，多租户
- 多种副本类型：平衡可用性、成本、性能
- 压缩：性能和成本平衡
- 硬件成本：SSD寿命及成本
- 具有弹性的可扩展性：大促



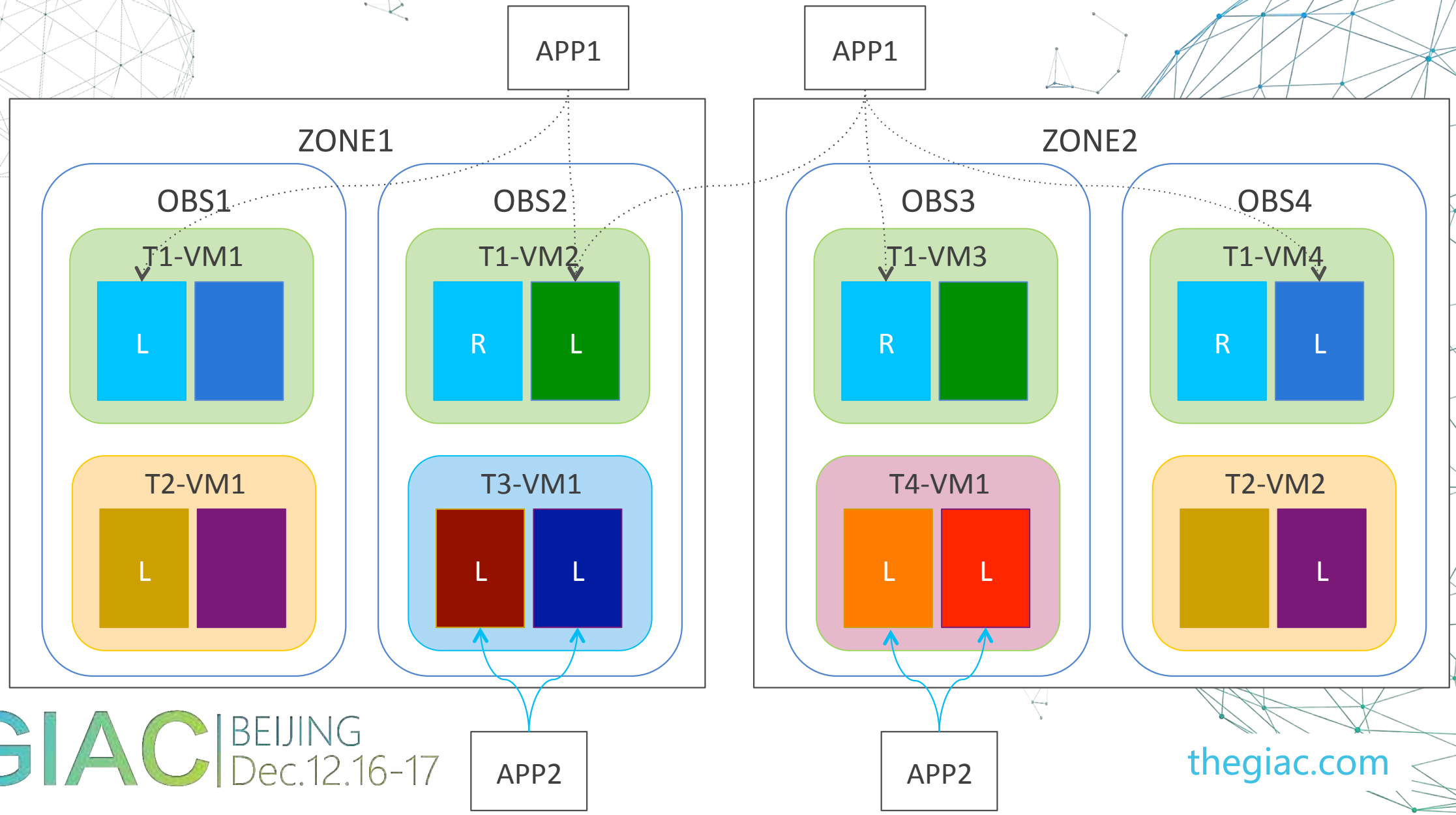
双11大促场景

- 短时间突发大流量
  - 短时间大量用户涌入
  - 一段时间（几秒、几分钟、半小时）后业务流量迅速回落
- OceanBase峰值性能稳定持久
  - 传统数据库：系统容量和峰值容量不匹配
  - OceanBase数据模型的优势
    - 写事务相当于内存数据库
    - 没有随机写对峰值时查询的影响
    - 批量合并平滑了峰值的写入压力

OceanBase大图1



OceanBase大图2

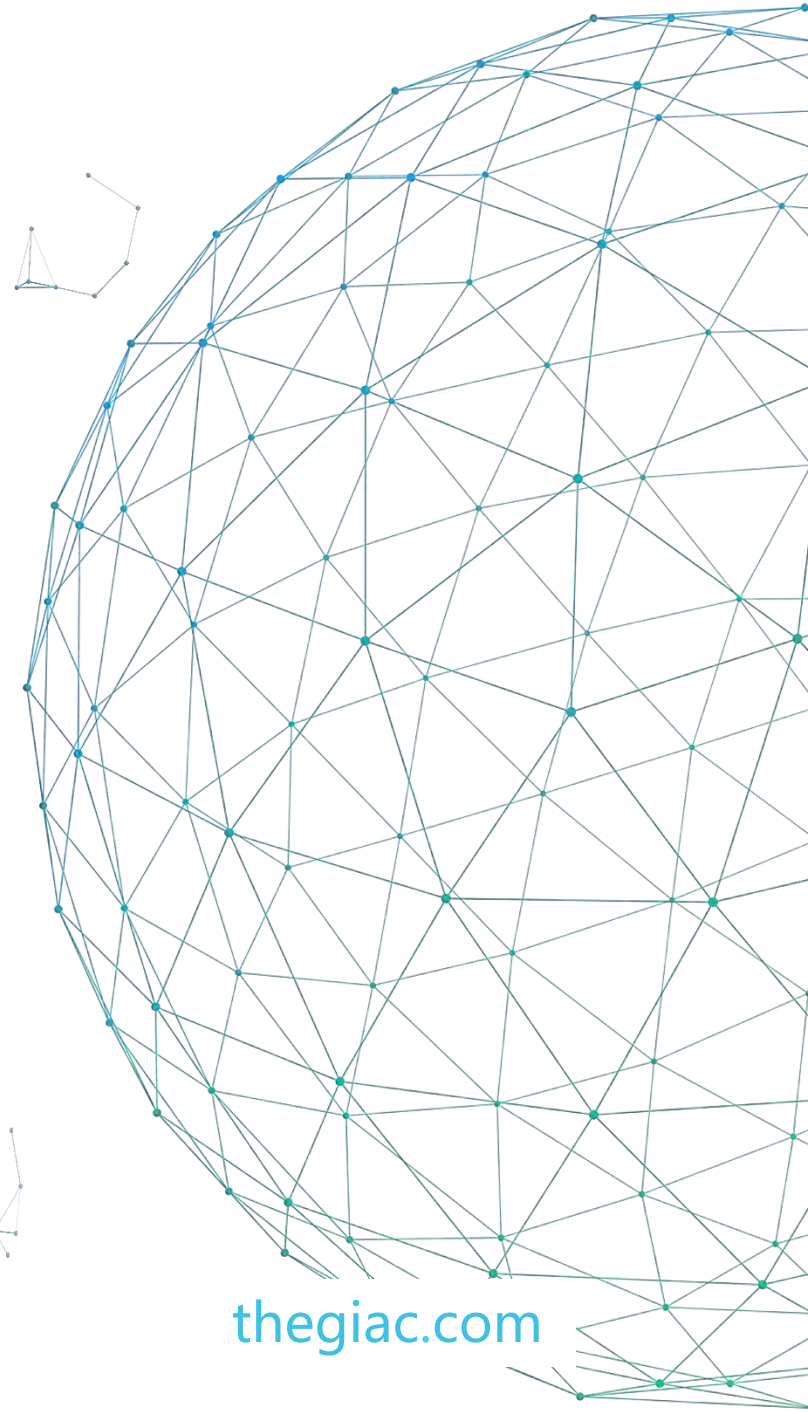


资源负载均衡

- 多租户负载均衡
  - 把租户的SQL-VM分配到集群内
  - 大租户和小租户适配
- 租户内负载均衡
  - 通过迁移副本和Leader平衡租户内各SQL-VM的负载
  - 最大化租户拥有资源的利用率
    - 避免潜在热点
    - 大分区和小分区适配
  - 考虑磁盘占用、IOPS、CPU利用率等多种资源

请求负载均衡

- 大查询及热点问题
  - 均衡更实时
- 非一致性SELECT查询
  - 本地化
    - 本ZONE > 本IDC > 本地域
  - 基于反馈的负载均衡
    - 本ZONE内多个只读副本，基于延时反馈选择

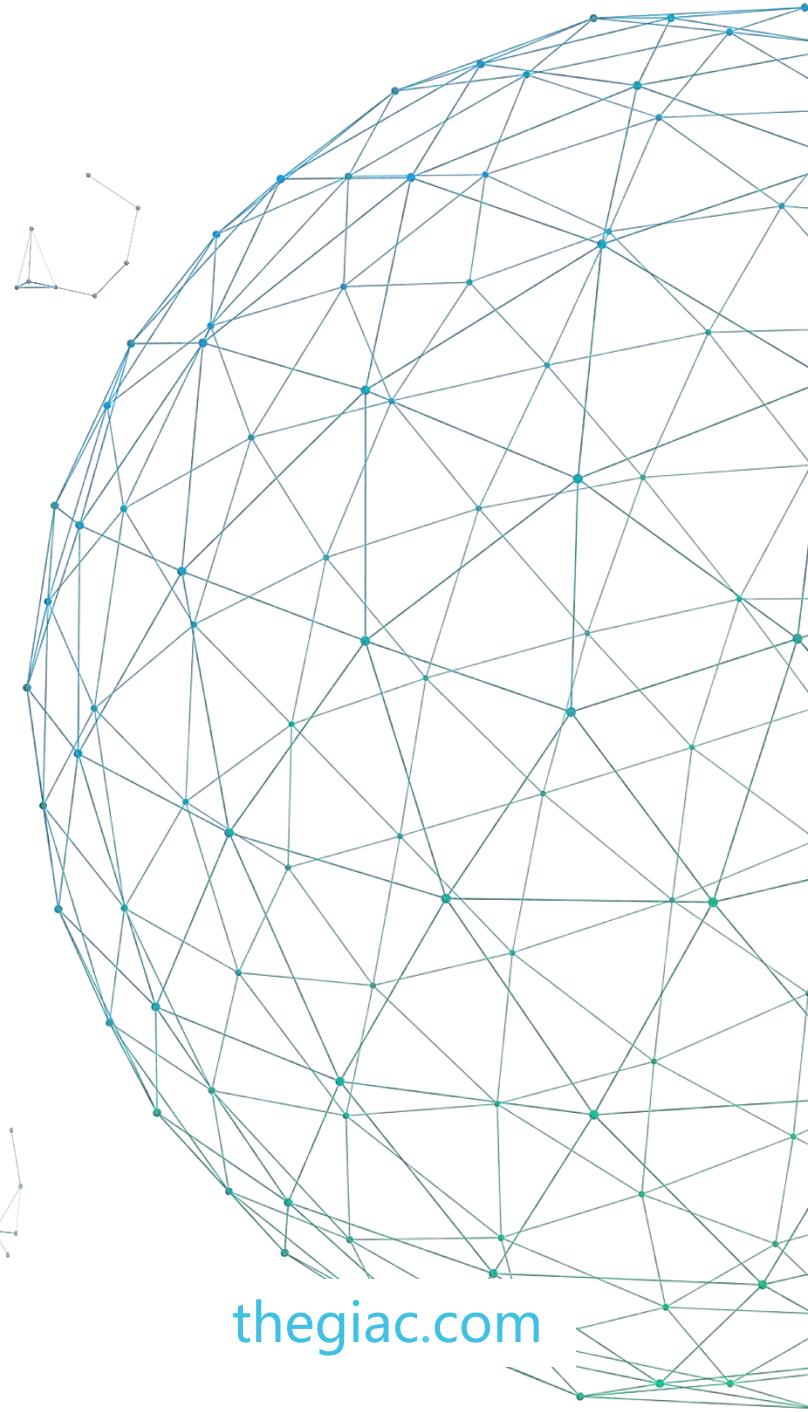


如何控制复杂度

- OceanBase系统实现的复杂度
  - 分布式存储
  - 分布式事务
  - 分布式计算（OLTP, OLAP）
  - 多租户DaaS

设计

- 抽象
  - 复用
    - 编码高度抽象
    - 模块复用：例如充分利用事务特性
  - 分层 vs. 效率
- 简单
  - 先正确，后优化
- 自校验
- 监控

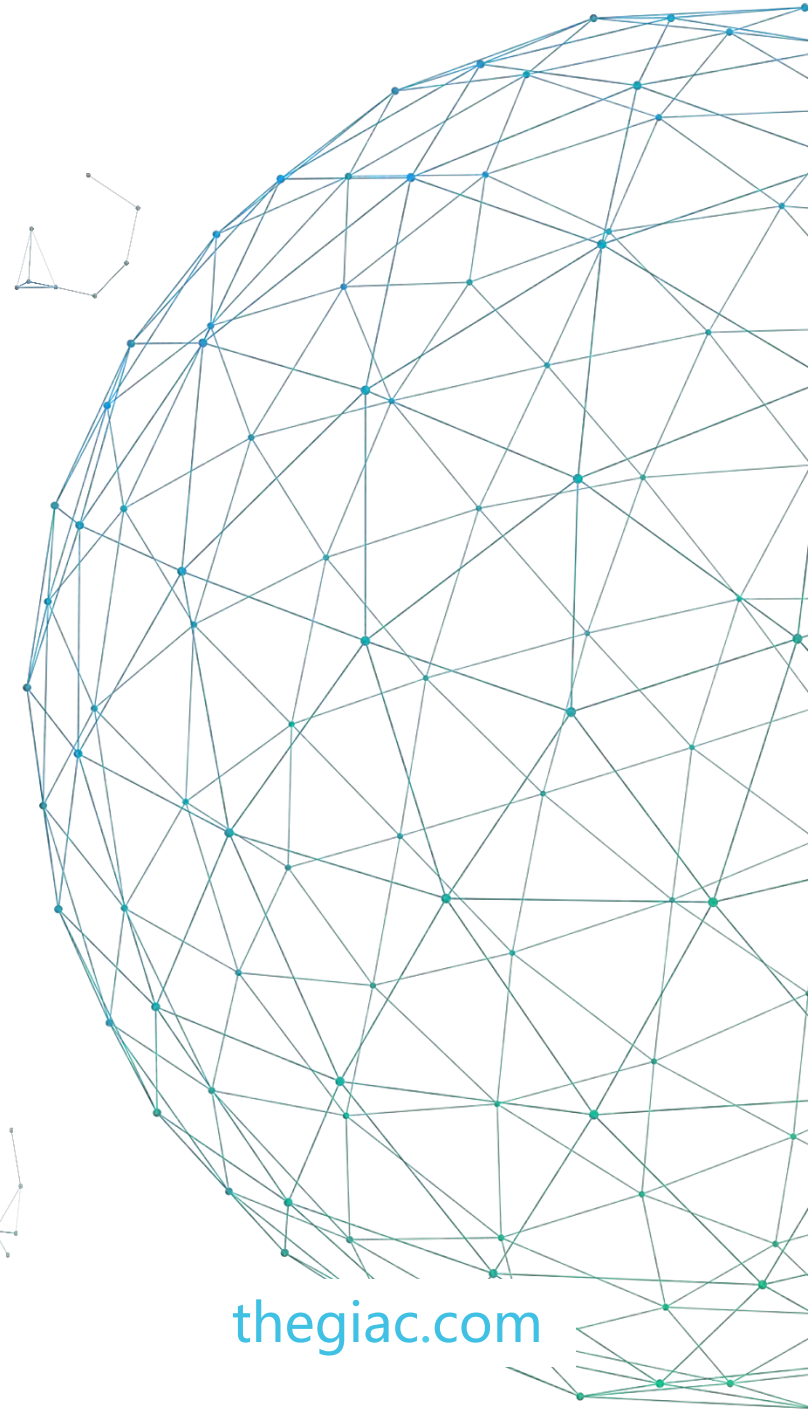


开发流程

- “严苛”的C++编码规范
  - 例如函数单出口，入参检查，指针判空，STL等
  - Coverity
- Code Review
- 准入测试
- 每日回归

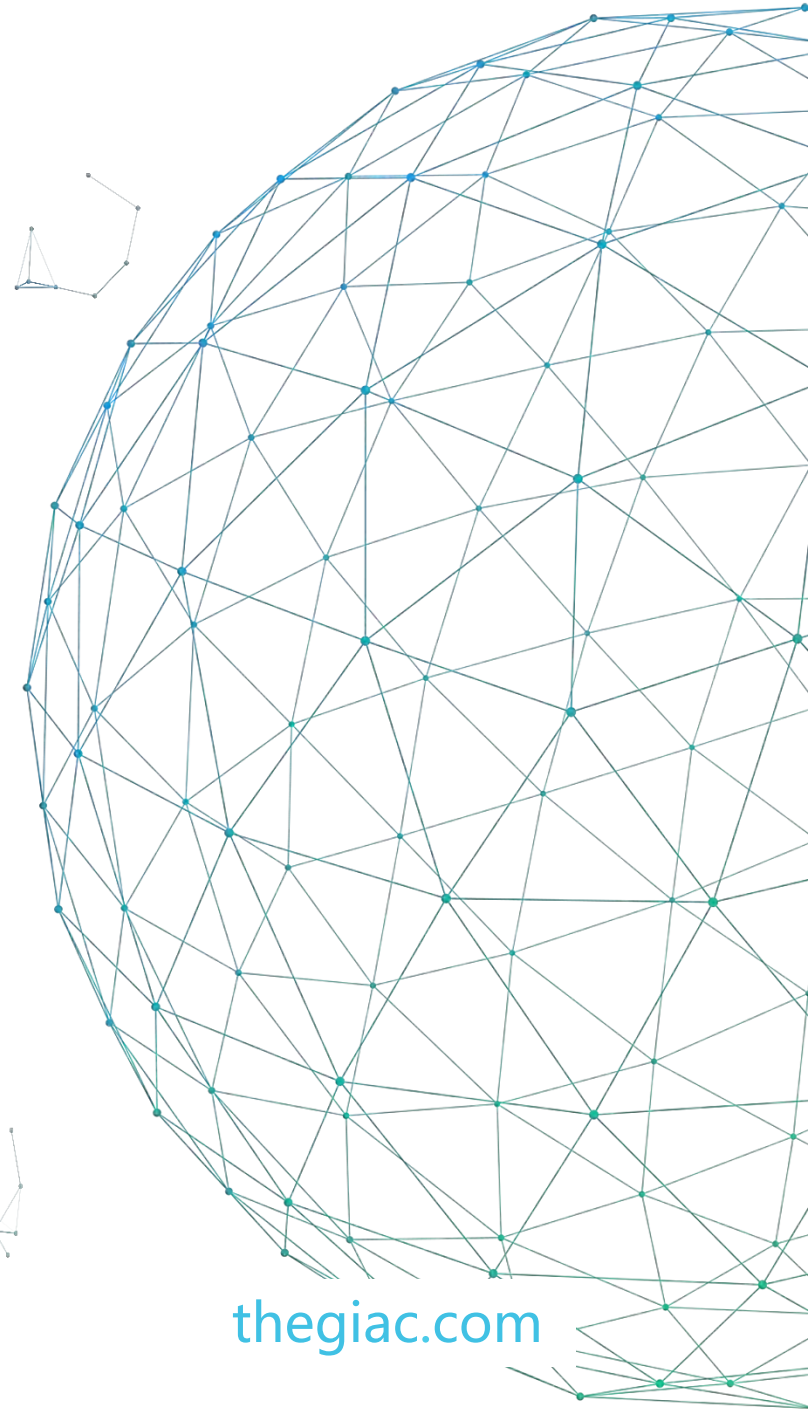
测试体系

- 单元测试：gtest，gmock
- 模块测试：SQL模块为例，obschema
- 功能测试mysqltest, RQG, pquery, obgene
  - mysqltest: PS, j-connector
- 非功能特性测试obtest
- 压力测试obmonstor
  - 错误注入
- 性能测试sysbench, obmeter
- 兼容性测试
- 事务正确性obstress
- 模拟业务测试obtrade，obbank



工具&效率

- git：主干开发+release分支
- ob-review：提交review并触发准入测试
- ob-pretest：智能单测运行
- ob-commit：提交前准入验证，关联bug
- fast-train：自动定位引起回退的提交
- ob-deploy：RD的瑞士军刀
- 分布式编译：dmucs+distcc
- HTCondor & ob-farm：测试作业调度
- AONE：bug, issue, review



GIAC BEIJING Dec.12.16-17

技术架构未来

架構 ARCHNOTES

Thank you!

- OceanBase竹翁
- 邮箱zhuweng.yzf@alipay.com

OceanBase团队诚招技术支持、质量保证、研发各类人才！

简历发送至：zhuweng.yzf@alipay.com