亿级海量数据的实时读写和复杂查询实践

--黄哲铿

《技术管理之巅》作者

“技术领导力社区”发起人

目录

多租户SAAS系统场景简介

- 数万租户的商家结算系统
- 数十万用户实时在线
- 每天产生亿级增量账单数据
- 实时读写，大量复杂SQL分页查询
- 月初每商家数百万明细数据导出量

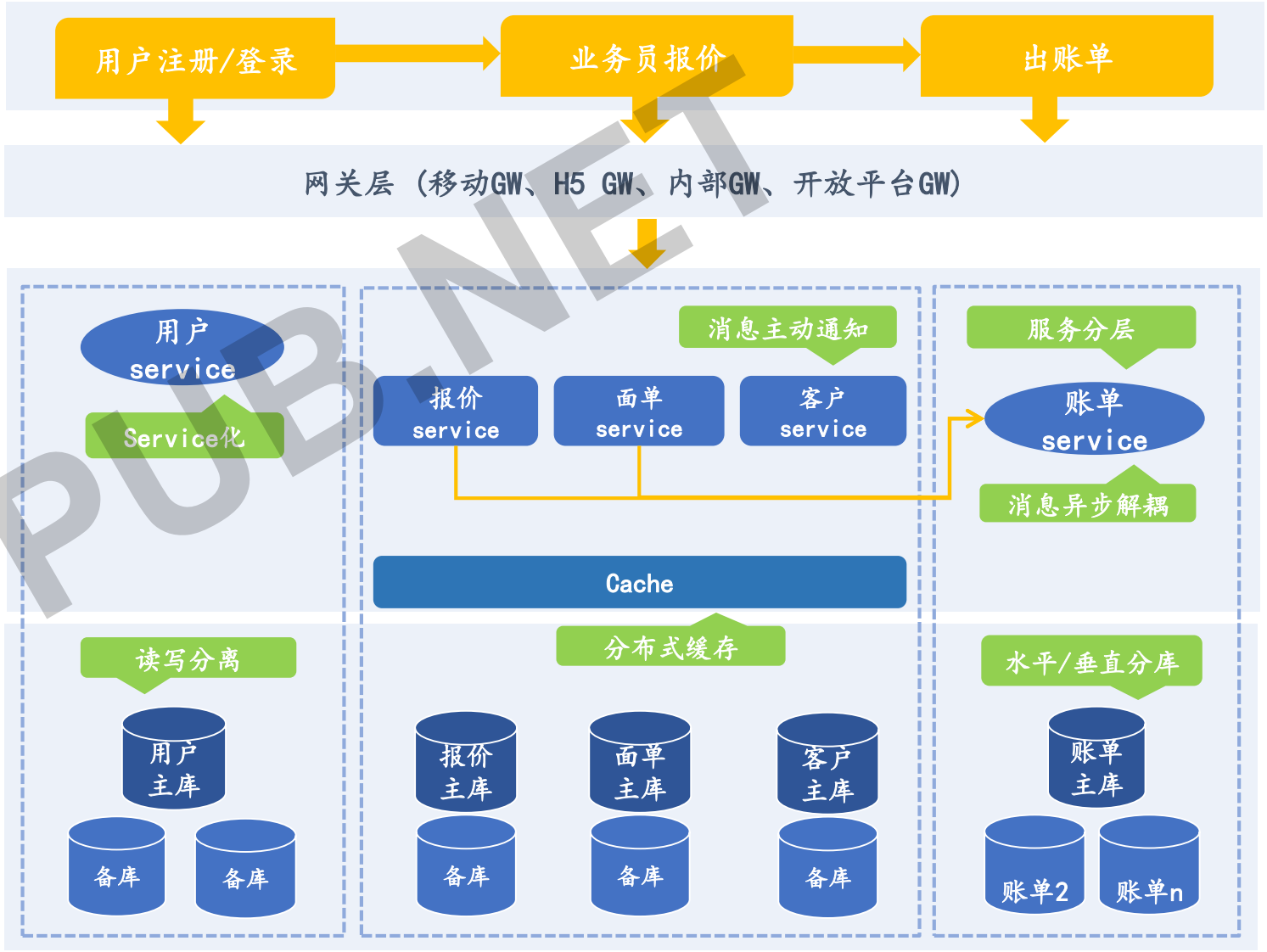
系统面临的挑战

- 亿级增量结算数据的存储
- 业务场景，对数据实时性、一致性、完整性要求高
- 数百个应用部署实例，对数据库造成的压力
- 结算系统的高可用性，要求99.95%以上
- 数据源来自消费，数据实时处理，且逻辑复杂度高

目录

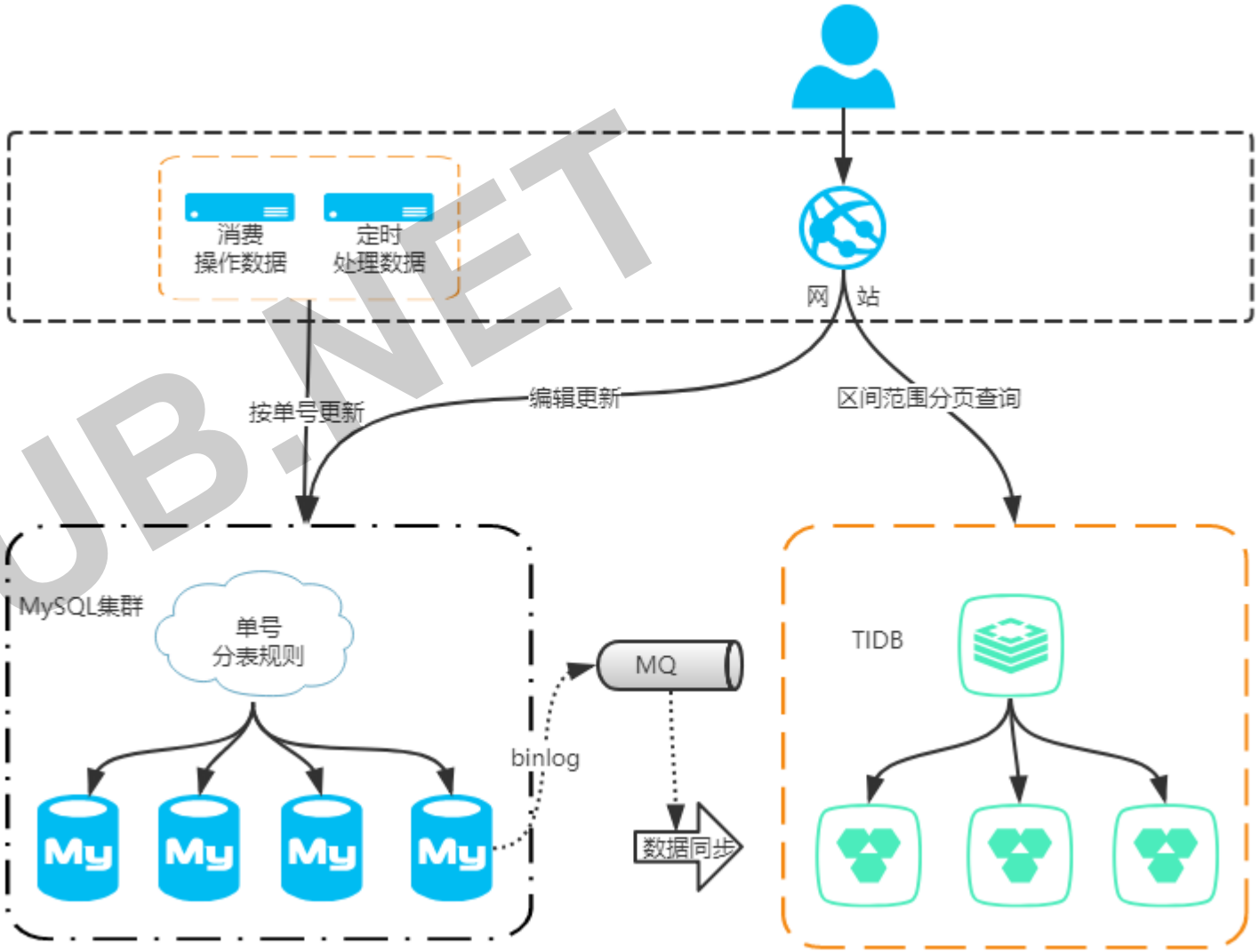
云原生微服务架构

- **高可用、解耦：**基于Spring boot、dubbo、 zookeeper， 自研网关、服务治理平台构建应用;
- **弹性扩容、易开发易维护**：基于Docker、K8S、openAPI的云原生微服务架构;
- **业务中台**：沉淀业务能力、以openAPI其它应用开放;
- **服务治理**：熔断、限流、鉴权、监控预警;



数据存储架构图

- 采用TBSchedule实现分布式job，将消费数据按分片规则，并发计算处理；
- 处理好的数据，存储在MySQL集群，主从模式，从库用于备份；
- 数据通过MQ同步到TiDB，作为只读库，不进行分表，以兼容现有程序；
- 3个月清理一次MySQL数据，半年清理一次TiDB数据



MySQL 分表分库策略

- 基于sharding-jdbc定制化的中间件，3种分表规则：商家ID、结算单号、时间
  - 商家操作自己结算数据的场景，按商家分表
  - 多个商家之间结算的场景，按结算单分表
  - 按时间周期统计的场景，按时间分表
- 以商家分表为例，通常多个小商家分在同一张表，超大商家独立分表，以免相互影响；
- 前期采用MySQL主从模式，读写分离；后期，读库转移到TiDB，且不分表；

数据异构与数据聚合

- 按商家ID、结算单号、日期，进行数据异构，用于不同业务场景的数据检索
- 基于canal的MySQL数据同步的异构方案
- 数据异构的存储方式可采用ES、REDIS等，根据不同业务场景决定
- 热数据、温数据、冷数据的分类别存储，如历史数据归档
- 数据聚合，应用采用了前后端分离，客户端需要合并请求、服务端做数据聚合

TiDB使用与优化

选型的思考：

- 解决 MySQL 的单机性能和容量无法线性和灵活扩展；
- 考虑到业务库以MySQL为主，选型必须协议兼容 MySQL；
- 强一致的分布式事务：事务可以跨分片、跨节点执行，并且强一致。

踩过的坑：

- 采用TiDB替代MySQL从库， TiDB不再分表，需解决各分表记录合并后，需解决主键冲突问题
- 切换顺序：离线业务 -> 非核心业务 -> 核心业务；
- 避免使用大宽表(<100字段)
- 对延时比较低的场景不太适合
- Delete 大量数据，GC 跟不上

不足及展望

- 使用 MySQL 作为存储，到了容量和性能瓶颈，业务容量翻5倍则难以支持；
- 分库分表查询，在中间层组合在架构上缺乏灵活性、可扩性
- 扩大TiDB的使用，为切换核心业务库做技术储备
- 对于 MySQL 、 TiDB、Hive、Spark使用场景的一体化整合；
- 大型互联网系统的架构的探索：云原生微服务+云原生数据库

微信号:

感谢聆听!

[技术领导力]订阅号: