微信支付

微信支付数据库管理实践

tomzhou

个人简介

- 组织：
  - 腾讯 - 微信事业群 - 技术架构部 - 运维中心
  - ID： tomzhou(周汤)
  - 2009加入腾讯，先后负责过拍拍、网购、易迅、充值、团购等业务的运维工作；
  - 2014.05转入微信，负责微信支付业务的运维工作；

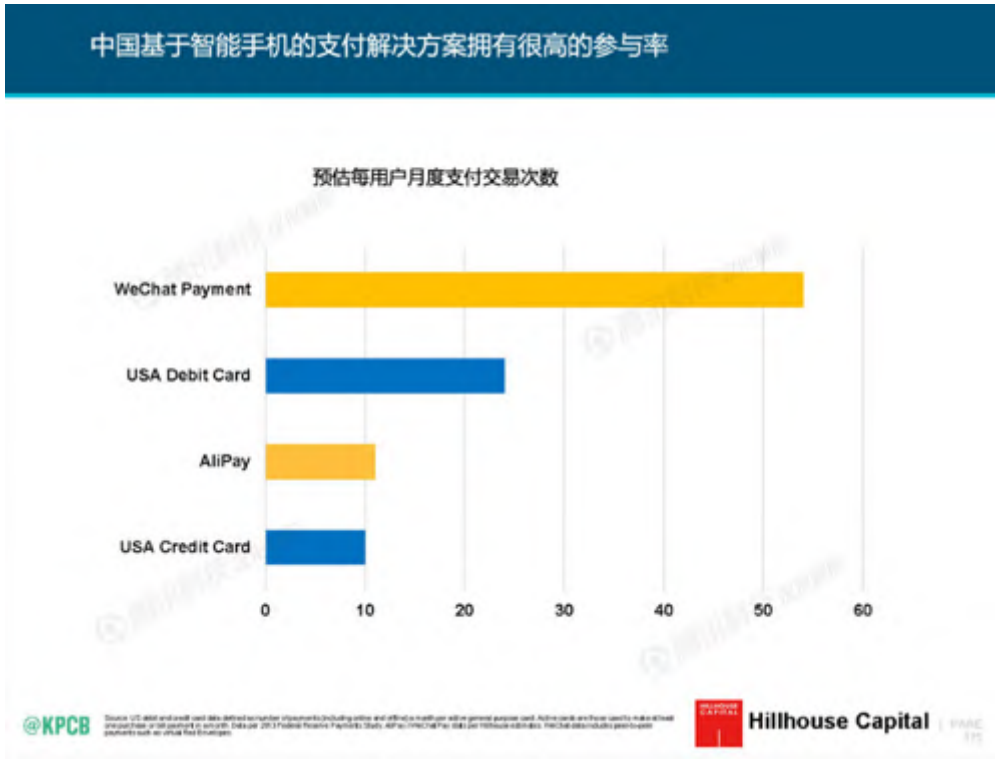
- 个人：
  - 开发、运维、数据库、安全都感兴趣；
  - 曾经的个人站长，博客： http://www.iamadmin.com/
  - 个人邮箱： admin.net [at] 163.com

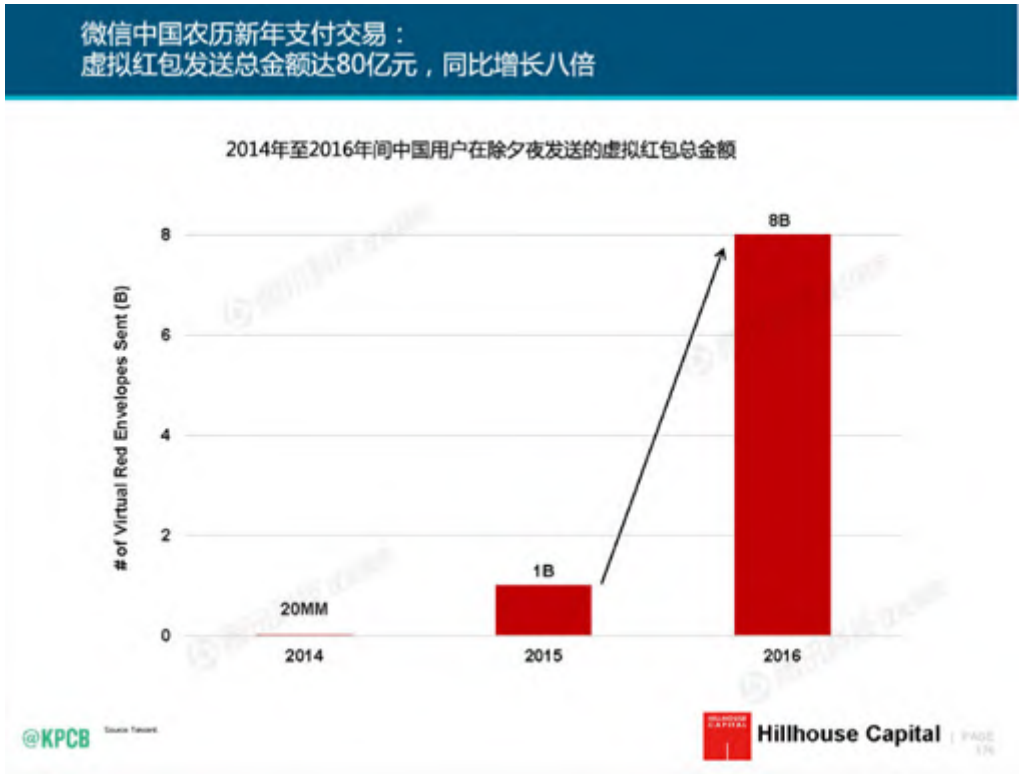
大纲

- 背景及挑战
- DBCMDB
- 变更
- 监控
- 安全
- 高可用
- Golang

背景 – 业务快速成长

- 移动支付时代全面到来；
  - 国内产品无论产品、用户量、支付频率都领先全球；
- 业界最大的支付和结算系统；
  - 支付峰值 15w/s
  - 入账 50w/s；





背景 – 数据库环境

- 基于PC + MySQL
  - PC服务器，SSD固态磁盘；
    - PC故障率 2%
  - 基于开源的MySQL，适当定制；
    - 无商业技术支持
  - 没有去IOE这个说法；

- 规模(微信支付)
  - 数据库服务器：1000+
  - 数据库实例：1000+
  - 数据库业务：500+
  - MySQL DBA：3

背景 - 目标

高性能	高可靠	高可用	安全

• 每秒几万支付；
• 数据库层面TPS达百万级别；
	• 数据强一致
	• 各种短款赔付场景
	  • 红包多拆；
	  • 多次支付；
	  • 支付状态错误；
		• 用户和商家零忍
		  • 7*24 高可用
		• 降级困难
		• 场景特殊
		  • 线下支付
		  • 医院
		  • …
			• 非法查询
			• 防篡改
			• 防丢失

背景 – 挑战

- PC硬件+开源软件；
- 每分钟百万支付，千万资金流水；
- 业务高速成长、快速迭代、变更频繁；
- 软硬件故障、运维不当，分分钟上百万的损失；DBA表示完全赔不起；

请计算DBA的压力

如何避免DBA跑路？

DBCMDB

数据库运维管理的基石

DBCMDB

- CMDB 的几个阶段

基础CMDB

- 设备、机架、固资、IP、责任人…
- IDC运维的基石

运维CMDB

- 服务名、端口、文件、定时任务、设备…
- 业务运维的基石

DB CMDB

- 业务、实例、端口、库、表、权限、责任人…
- 数据库运维的基石

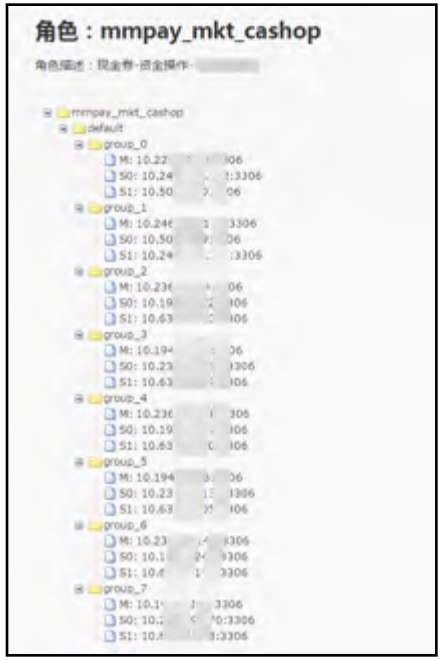
DBCMDB – 为什么？

- 业务众多、配置、部署错综复杂
  - 几百个业务、上千台服务器、上千个数据库实例；
  - 单业务多组、单机多实例，单实例多库部署；
  - 分组、分库、分表部署；
    - 各种命名规则；
    - 各种编号规则；

- 问题：
  - 一个普通表结构变更，如何准备定位到哪些机器、哪些实例、哪些库（库名）、哪些表（表名）需要变更？
  - 部署经常会有调整，如何确保调整后的变更、监控、备份等操作同步生效？
  - 工作上没法传承，新同学加入就懵了；

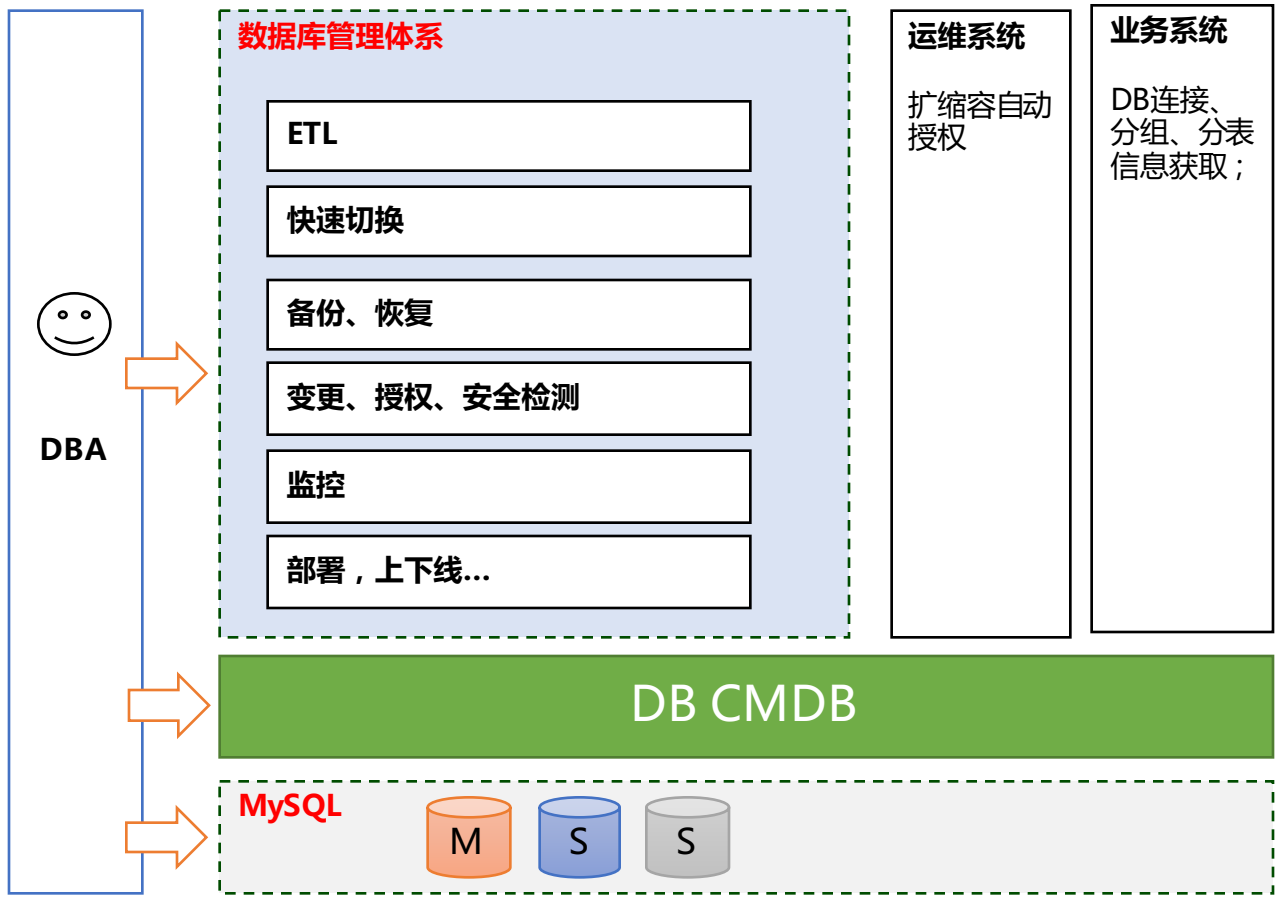
DBCMDB - 怎么做?

- 核心数据
  - 基础特性：IP、Port、Version、MySQL基本配置...
  - 业务特性：业务、字符集、分库数量和命名、关联实例、分表数量和命名、主从关系、干系人...
- 工具
  - Web界面管理，集中管理、统一引用 、开放接口；



DBCMDB – 能做啥？

- 数据库运维质量和管理的重要支撑；
- 以DB CMDB为中心建设工具
  - 全局配置中心



DB变更

如何高效、高质量的变更？

DB变更

- 各种变更
  - 部署变更：主从切换、扩缩容...
  - 字典变更：表、字段、索引...
  - 业务变更：架构调整、数据搬迁...

- 互联网的迭代速度，与金融的稳定性如何兼得？

DB变更 - 以前

- 开发直上数据库

- 优点：
  - 简单、高效、直接；

- 问题：
  - 专业度不够
    - 未预估数据量和制定分库分表规则；
    - 命名混乱；
    - 索引不当；
    - 字符集、引擎；
    - 触发器、外键；
    - ...
  - 变更故障频发
    - 加个索引搞挂系统
    - 误操作

DB变更 – 后来

- 变更需求系统
  - 所有变更走系统，可评审、审计、回溯；
  - 开发提单、DBA审批；
  - DBA手动执行；

- 优点：
  - 专业度、DB稳定性、可靠性、性能提到了显著提升；

- 问题：
  - 沟通成本高；
  - 效率低，瓶颈在DBA；
  - 对DBA要求高，配置容易出错；
  - 变更怎样减少对DB性能、业务的影响；

DB变更 - 现在

- DB变更系统
  - 开发自助提需求；
  - DBA审批、制定执行计划；
    - 技术评估；
    - 灰度，串行、间隔；
    - 定时，业务低峰时段自动执行；
  - 基于DBCMDB，自动执行并反馈结果；
    - 找出目标MySQL IP；
    - 找出目录库、表，以及命名、分组规则；
    - 自动拼装SQL；
    - 初级语法检查；

- 优点：
  - 高效、专业、可靠

DB监控

数据库仪表盘

监控 - 以前

- Nagios, Zabbix

- 优点：
  - 监控项较为全面
  - 功能较为完备

- 问题：
  - 配置管理不便，上千个实例的添加和同步变更；
    - 配置遗漏；
    - 长期误屏蔽；
  - 告警策略简单；
    - 逐IP配置；角色变换、业务变换、上下线容易配置错误；
  - 缺乏告警后的事件处理流程；
    - 曾出现过严重事故；
  - 操作和界面不够友好；

监控 - 现在

- 自建监控系统和事件管理系统

- 优点：
  - 基于DB CMDB，部署调整同步更新监控，无需人工介入；
  - 基于角色，模板化策略配置，极大提升管理效率；
    - M/S可能需要配置不同的告警阀值；
    - 根据业务重要度、敏感度制定模板化的策略；
    - 业务与策略模板绑定，部署调整无需调策略；
  - 多终端（WEB、微信），随时随时关注DB健康度；
  - 完备的事件处理流程；
    - 预警、告警分级处理；
    - 按业务、角色智能收敛；
    - 事件升级处理策略；
      - 5分钟内响应；
      - 30分钟内处理完；

数据安全

DB授权

数据安全 – 整体

- 主机
  - 专有跳板机登录；
  - Shell操作审计；
  - 敏感操作预警；
- 监控
  - 异常连接监控；
- 授权
  - 最小化原则，按需申请审批；
  - 精准授权；
- 业务
  - 关键字段脱敏、防篡改；
  - 安全扫描、分层设计防SQL注入；
- 备份
  - 加密存储；
- 规范
  - 严禁任何人员非法查询、修改现网数据；

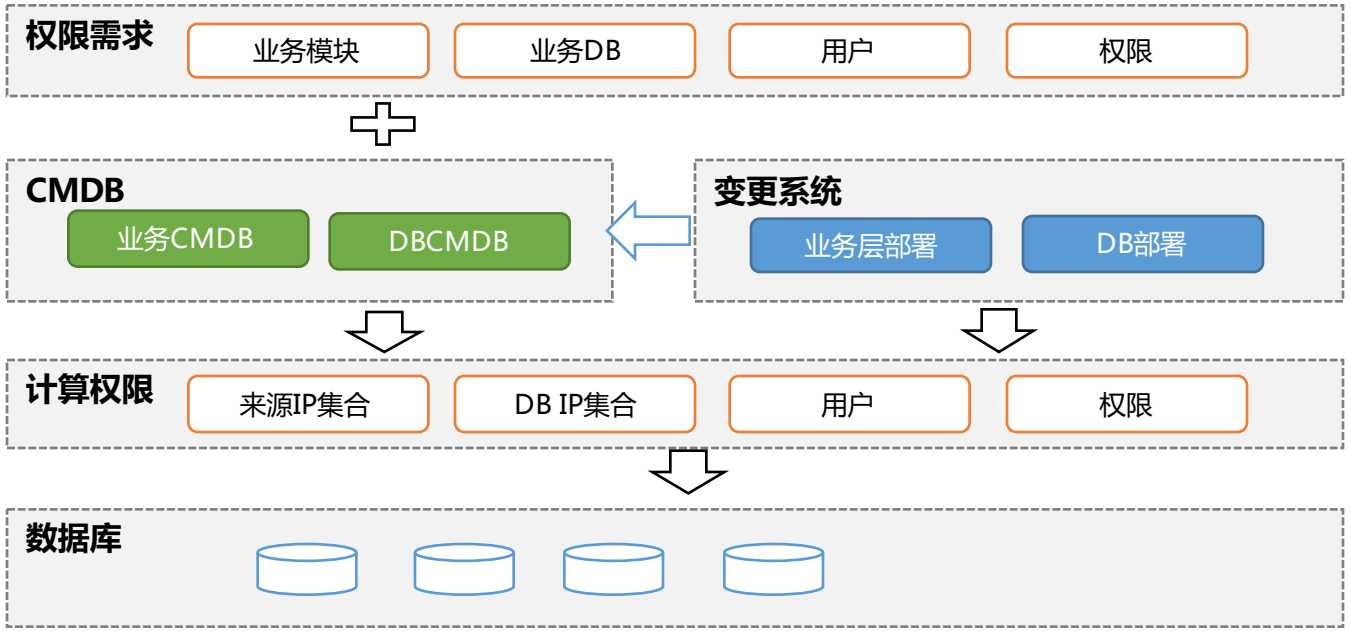
数据安全 – 授权系统

- 以前
  - 开发提需求、DBA逐IP手动执行；

- 主要问题：
  - 效率低下，每次上线、扩容都需要操作；
  - 权限容易遗漏；
  - 容易出错，扩容时提供的密码与原密码不同；
  - 只管授权，没有回收，权限被放大；

数据安全 – 授权系统

- 现在：
  - 基于业务模块+业务DB来管理权限；
    - 系统自动根据CMDB换成 IP来授权；
    - 部署调整无需手动变更权限；
  - 自助流程（开发提单，运维审批，自动执行）；
  - 过期权限，废弃权限自动回收；
  - 与发布变更系统整合，扩缩容自动授权和回收权限；

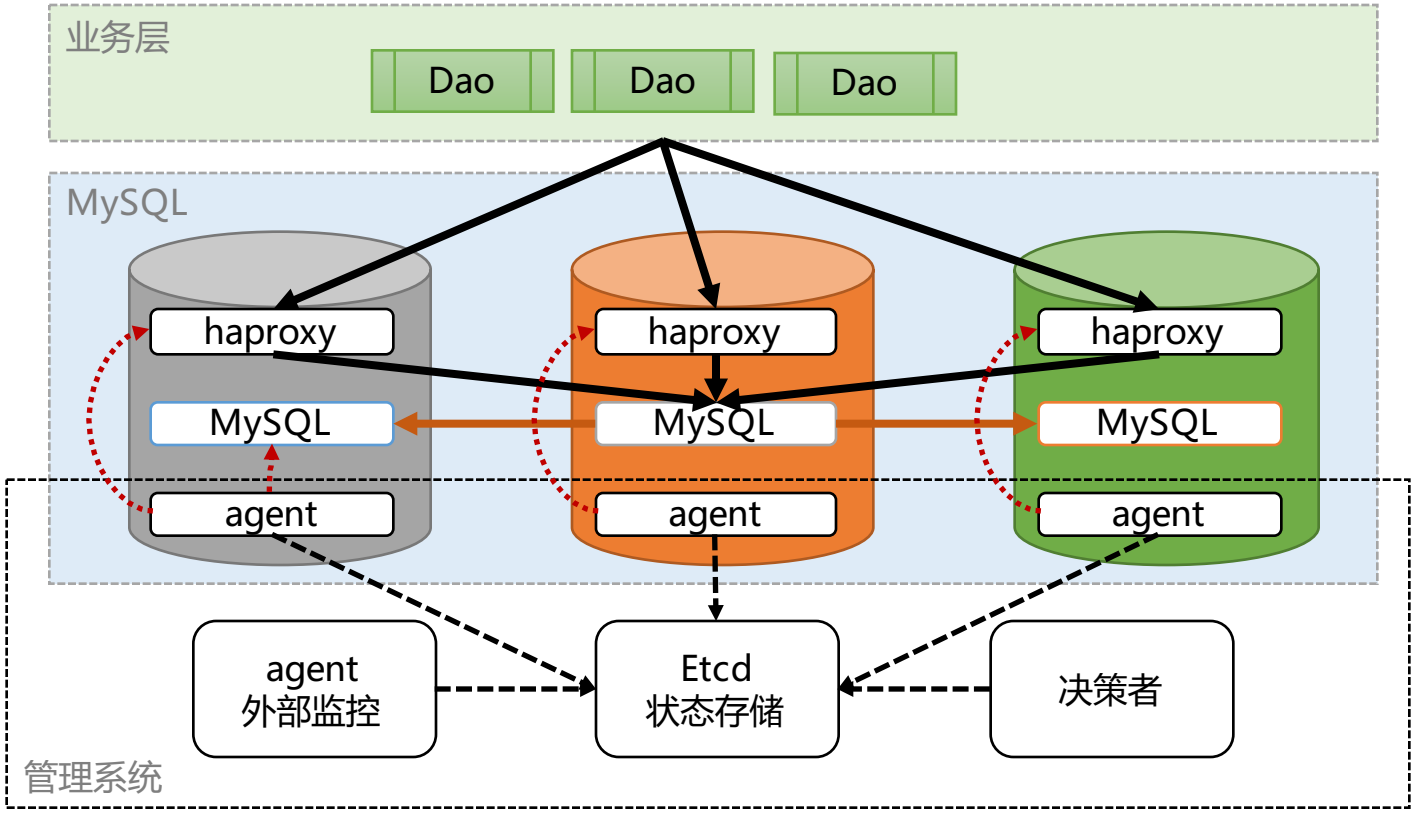


高可用

MySQL + 业务双层自动切换

高可用 - 数据库

- 主从自动切换
  - 纯粹的快速切换工具；
  - 完全兼容现有部署、运维方式；
  - 轻便的升级、回退方式；
  - 三机读写、三园区容灾；
  - 稳定、可靠；

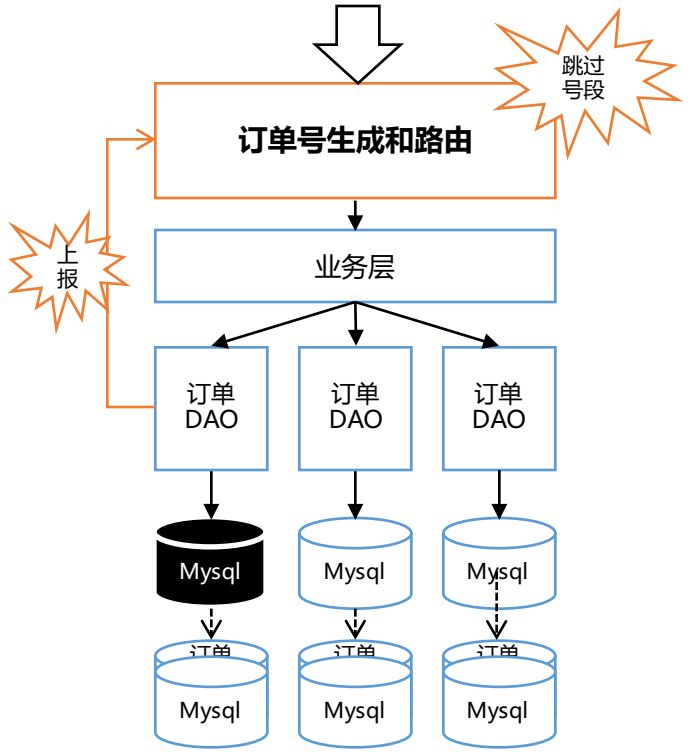


- PhxSQL
  - 强一致性；
  - 快速切换；

高可用 – 业务

- 自动跳单
  - 只适用于单号内部生成，根据单号路由到DB的业务；

  - 数据库多组部署；
  - 业务探测并上报DB健康状况；
  - 达到失败阀值，生成订单时自动跳过此组；
  - 重新点火恢复服务；



Golang

下一代互联网开发语言

Golang – 下一代开发语言

- 简介：
  - 来自Google；
  - 语法简洁、清晰、高效；
  - 高性能，跨平台；
  - 语言内置并发特性充分利用多核CPU；
  - 自动垃圾回收；
  - 强类型、静态编译、内置高级语言类型；
  - 本地程序，无需虚拟机，部署极简；
  - 内置大量网络通信、加密、解压缩、数据库等模块，显著提升开发效率；

  - 拥有C/C++级别的性能、稳定性、部署便捷性，以及拥有3P、Java等语言的易用性和高级语言特性；

  - 分布式后台程序首选！
  - 高性能、代码保护的ＷＥＢ开发首选！
  - 运维开发利器；

Golang – 应用

- 业界
  - Docker
  - Google Kebernetes
  - Etcd
  - Nsq
  - Hyperledger
  - …

- 微信支付运维
  - 外网质量监控
  - DB监控
  - DB快速切换
  - …

谢谢