GOPS2016
Beijing

全球运维大会

2016

DevOps 2.0：重塑运维价值

会议时间：12月16日 - 12月17日

会议地点：北京国际会议中心

主办单位：高效运维社区 GreatOPS Community　开放运维联盟 OOPSA Open OPS Alliance

微信支付数据库管理实践

莫晓东　微信支付DBA

目录

存储层方案和设备选型

- 项目挑战：
  - 春节红包节日量级是日常的100倍。
  - 无法借鉴、摸着石头过河。
  - 精确压测性能，为容量评估和限流提供依据。
  - 从配置、部署、容灾三方面深入优化，为业务保驾护航。
- 是否继续使用MySQL？
  - 需要多少机器，怎样部署。
  - 使用什么机器类型。
  - 可能出现什么问题，怎么解决。

继续使用MySQL

- MySQL支持事物，满足一致性要求。
- 结构化存储，紧凑、连续。
- 支持多索引。
- 部署简单，工具支持。
- 团队技术积累。

- 设备改进。

  尽量使用 PCIE-SSD，如fusion-io。

  sas使用硬raid卡，BBU&Cache，WB，Raid10。

  主板bios关闭节能、最大性能。

  更好的CPU、内存、网络...

•调整系统参数。

  文件系统ext3 ext4 xfs。

  磁盘挂载优化 nobarrier noatime。

  系统内核定制。

  Io scheduler、numa、swappiness、drop。

MySQL优化

- MySQL版本选择。

  percona sever 5.6（线程池、4k page）。

  升级到5.7。

- 部署优化。
  - 多实例，交错部署。
- MySQL参数优化。
- 定制优化。

参数设置。

关闭Query Cache。

innodb_file_per_table = 1。

innodb_io_capacity = 5000。

innodb_max_ditry_pages_pct = 30。

innodb_flush_neighbors = 0。

innodb_random_read_ahead = 0。

sync_binlog = 1。

innodb_support_xz = 1。

innodb_flush_log_at_trx_commit = 1。

innodb_flush_method = O_DIRECT_NO_FSYNC。

back_log = 120。

metadata_locks_hash_instances = 16。

table_open_cache_instances = 16。

目录

规范制定

- 使用规范。
  - 计算移到逻辑层。
  - 平衡范式和冗余
- 命名规范。
  - 建议小写，统一分隔符。
  - 不同的前缀、高区分度。
- 字段规范。
  - 合适的字段。
  - Null属性。
- 语句规范。
  - 尽量简单。
  - 测试和审核。

- 索引规范。
  - 控制索引数量。
  - 主键紧凑，使用数字。
  - 前缀索引。
  - 不使用外键。

- 默认字符集从utf8改latin1。

库表设计

- 字段缩减，去冗余。
  - 冗余字段精简。
  - 单号精简。
  - 字段类型精简。
- 按天分表，循环使用。
  - 确保数据单表数据不会无限增长。
  - 用truncate代替delete清理。
- 垂直分表力度更细。
  - 小表性能更佳。
- 主键从单号修改为自增字段。
  - 自增主键减少体积，加速插入速度。

- 不同的内容，不同的存储。
  - 有些内容使用kv更适合。
  - 用户维度迁入列表服务。
  - 扩散读和更依赖cache。
- 业务拆分：个人红包和企业红包分离。
  - 企业红包特有字段去除。

语句优化

- 不在数据库做运算，不在业务SQL使用函数。
- 控制单表数据量，控制字段数量，控制索引数量。
- 不要依赖外键和触发器。
- 索引选择率不高，索引不良。
- 缺少主键和索引。
- 拒绝大事物、慎用复杂查询。

- 语句优化。
  - 不查不需要的字段。
  - 非核心查询移到从机。
  - 大操作合并查询。

禁止使用order by rand() !!!
语句简单化，大语句拆成小语句。
事务简短，禁止长时间事务。
线上业务语句，不使用函数和运算。
指定字段，而不是select *。
or子句使用in(...)或者union代替。
limit避免数字过大，翻页过多使用range语句代替。
慎用join连接，经常join可以适当冗余；但单表字段过20时join更划算。
比较子句，请使用同类型，如果是字符串请保证同编码。
导入数据时，使用insert values(),...()或者load，避免单条插入。
集中的修改操作请打散。
尽量不用not in，有坑。in的数据要少。
like子句，%号不要放在条件左边。
避免单条语句更新多个表。。

目录

有损服务、柔性可用、大系统小做

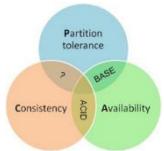
- 什么是有损服务？有损服务是通过精心拆分产品流程，选择性牺牲一部分数据一致性和完整性从而保证核心功能绝大多数运行。核心功能是摇，拆，发。其它模块异常立即降级防止引起雪崩。这里用春节摇一摇红包做例子。
  - 过载保护，层层过滤、快速拒绝，把千万级别的请求减少到万级，减少DB负载。
  - 异步处理，耗时最长的入账操作，直接跳过，异步处理。
  - 柔性策略，关闭不重要的功能模块，比如查看收发历史、关闭祝福语。
  - 资源隔离，拆分红包DB为数十个Set，出现故障只影响数十分之一。
  - 大系统小做，功能单一。逻辑层需要减少模块耦合，存储层也需要多源。
  - 故障处理，如果DB故障直接替换空白机器，未领取的红包锁定退款。

故障预案，可控 ；未算胜先算拜，极端情况。

高可用

- 业务最终一致性，cap、base。
  - 允许丢失和对账、异步达成。
- 金融业务的特性。
  - 事物一致性，oltp、acid。
  - 可靠性高，99.99%。
- 大规模系统的问题。
  - 单台设备故障导致整体不可用。
  - 服务关联性，互相影响故障扩散。

- 基本可用Base。
  - 放弃每个步骤对事务的依赖。
  - 充分理解和细分业务需求。
  - 针对用户场景精心设计。
  - 核心事物的保证。



隔离和拆分

- 业务必须在控制下运行。
  - 万有一失，用户重试。
  - 伸缩调度，降级服务。
  - 基于对用户需求、行为分析的细致分析理解。

- 分离方式。
  - 轻重分离。
  - 部署分离。
  - 资源分离。
  - 用户分离。

- 限流算法。
  - 计数器。
  - 漏桶算法。
  - 令牌桶算法。

目录

红包野蛮生长

- 红包使用量急剧增长。
  - 拆事务性能瓶颈，高峰抖动。
  - 主从同步压力，对账不及时。
  - 存储容量压力，设备已缩容。

月增速翻倍并保持
运维和开发处于高负荷状态
业务抖动多次扩容
需要进行业务重构

- 需要进行业务重构。
  - 运维和开发处于高负荷状态。
  - 先应付眼前。

- 性能同样需要优化，拆红包单组DB拆峰值仅仅2000+，预计春节10w峰值，也无法完成任务。
  - 红包系统的性能瓶颈在拆事务上。

异步和Cache

- 事物控制。
  - 事物语句精简和优化。
  - 入账分布式事务。
  - 提高并发度，请求排队机制。

- 异步化。
  - 异步入账。
  - 异步用户信息。
  - 异步补拆，异步入账。

- 先查cache。
  - 通过cache判断能不能抢。
  - 缓存重复查询的内容。
  - Memset ~1.5M/s ，ram顺序读 ~4G/s

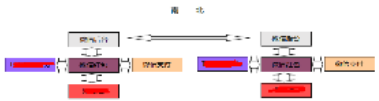
- 流量控制。
  - CGI无状态
  - 资源静态化
  - 业务流程异步化
  - 过载保护
  - 多级读缓存
  - 订单写缓存。

•梳理主机SQL，索引优化、语句优化、请求精简。

•高峰期抖动，优化请求时间分布。

异地多活

- 仍以红包的异地多中心为例子。
- 挑战。
  - 就近接入。
  - 城域容灾。
- 南北分布设计。
  - 系统切分，相互独立。
  - 流量可调控。
  - 相互容灾。
- 收益。
  - 故障恢复。
  - 流量均衡。
  - 降低时耗。

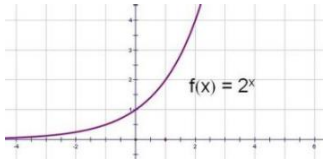


- 红包系统只部署深圳。
  - 跨城流量高延时。
  - 深圳机架位不足。
- DB宕机后的数据安全
  - 主备切换风险：主备延时，数据不一致。
  - 切换后，数据写脏。
  - 资金风险。

目录

空间爆增带来的重构需求

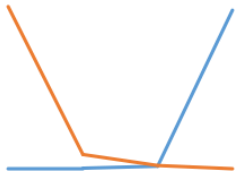
- 表空间回收困难，现网库删除慢，历史库存储过大。
- 字段过冗余，如果按每月约翻倍的速度增长，到年底是很惊人的。
  - 假设到年底还有6个月，如果2^6=64倍，那到2月份过年2^8=256倍。
  - 目前大约占了20T历史数据，如果几十几百倍，机器数量，业务成本无法承受。



- Innodb压缩表。
  - 使用InnoDB压缩格式，为冷热分离争取时间 。
  - 压缩率订单库44%，用户库57%。
  - 可接受的性能下降10-15%。
- tokudb压缩表。
  - 性能问题抖动。
  - 高达6-12倍的压缩率。

冷热分离

- 分离依据 。
  - 分析数据访问情况占比， 实测订单3天内，访问占比98.9%。
- 优化目标。
  - 提高性能，减少现网库表数据量。
  - 节省成本，历史数据使用低成本大容量设备。
- 历史库方案。
  - 按日分表，proxy路由请求。低峰时段搬迁数据，现网只保留7天。
  - 改善性能，从raid10+tokudb迁移到NoRaid+myisam，解决迁移问题和优化查询性能。



暂时解决问题，脱离容量问题，性能稳定性增加。

字段控制更重要，核心字段可能只占1/10的空间。
比字段控制更有效的是业务上进行优化，直接去掉这块数据，釜底抽薪。

运维成本控制

- 合理的运维目标和SLA。
  - SLA指标的提升是否有价值。
  - 符合业务的发展阶段。
  - 区分业务内部不同模块。
- 质量控制和操作标准化。
  - 人员操作失误是主要的故障来源。
  - 按计划变更、重要操作review。
  - 系统代替人来判断，灰度必要性。
  - 知识库建设和流程建设。
  - 完善监控和备份。
- 服务器分类。
  - 使用不同类型的存储应用，DB、KV。
  - 最大化资源利用，合理的运营规划。
  - 计算型、存储型、IO密集型、大内存型。
  - 统一监控管理。
- 生命周期。
  - 预算管理。
  - 信赖迭代、不做提前工作。
  - 从上线到下线。
  - 复用，时间互补和空间互补。
  - 云服务能力。

目录

DevOpsDays 即将首次登陆中国

DevOps 之父 Patrick Debois 与您相约

DevOpsDays 北京站 2017年3月18日

门票早鸟价仅限前100名，请从速哟

http://2017-beijing.devopsdayschina.org/

想第一时间看到
高效运维社区公众号
的好文章吗？

请打开高效运维社区公众号，点击右上角小人，如右侧所示设置就好

Thanks

高效运维社区

开放运维联盟

荣誉出品