一个“兼职 DBA” 的数据库运维经验谈

小米科技 谢良

xieliang@xiaomi.com

一个“兼职 DBA” 的数据库运维经验谈

- 关于我 (weibo.com/bestxieliang)

  06 年毕业，曾服役过金山、 *Oracle* 、淘宝， *2011* 年初加入小米，司职米聊服务端开发码农，同时兼职负责维护米聊的数据库和一些公司安全事宜，目前正参与开发海量小文件分布式存储系统 *MFS(* 类似淘宝的 *TFS)*

  小米在过去一年一直没有一个专职 *DBA* ，碰巧之前偶在 *Oracle* 和淘宝喜欢学习数据库，所以”被迫”成了兼职 *DBA* 。。。

  哦，对了还有，我不会写 *PPT* 也不太会表达 *~!@#$%^*

一个“兼职 DBA” 的数据库运维经验谈

- 米聊数据库运维现状

- 没有专门时间去钻研 & 测试，所以细节还没做到位
- 沿着业界大厂的方向走，求稳，还没开始求创新
- 没一个专职正规的 *DBA* ，有些地方较山寨 *:)*
- 报警监控粒度比较”豪放”，没个顺手的监控系统

一个“兼职 DBA” 的数据库运维经验谈

- 米聊数据库运维现状
- 线上数据库版本：官方 *5.1* + *Percona 5.5*
- 十台服务器：包括了用户关系系统和离线消息系统的两台异地灾备服务器；真正的核心业务主库用了三台
- 从库服务器大部分都部署了两个从库实例拉不同的主库
- 为了省成本，主库曾部署过其它服务，比如 *zookeeper* 服务端、离线分析 *job* 甚至是一些 *java* 中间层服务，现在少了。。。
- 备份用的 *xtrabackup*

一个“兼职 DBA” 的数据库运维经验谈

- 米聊数据库运维现状
- 几个主库都不大： *160G* 、 *70G* 、 *30G*
- 单表大部分有几百万到一两千万记录
- 单表多的有 *6000w+* 行，大的有 *30G+*
- 老业务大都单库十表，新的单库百表
- 程序架构对分库分表已支持，目前没瓶颈，没分库
- *HP380G7 + DELL R710*
- 统一 *SAS 15k* 转硬盘， *RAID 10*

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- *2011* 年 *3* 月米聊几千人同时在线，注册用户十几万；
- *Master/slave* 两台数据库服务器，当时从库只起了准实时热备的作用，读流量没打上去
- 一个月后，随着业务发展， *db io util%* 稳定 *100%* 。。。
- 调查分析：用的官方原版 *mysql5.1.41* ，业务表是 *innodb* 引擎，但几乎所有参数均是默认值，包括 *innodb_buffer_pool_size* 参数。。。
- 当时服务器内存是 *24G*

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- 于是调大 *bp* 值到 *12G*
- 同时， *innodb_flush_log_at_trx_commit = 2*

- 效果： *bp* 命中率从 *95%* 变成了 *99.96%*

  io %util 基本在 *10%~20%* ，偶尔上到百分三十

这时，米聊在线用户上万了，撑过来了。。。

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- 那时每天在线用户数每天涨 *10%*，想想复利效益。。。
- 没过几天，两三万人在线了， *db io %util* 基本又到 *50%~90%*
- 继续调查：发现磁盘的 *await/svctm* 值高，经常在 *xx* 毫秒级
- 开始怀疑 *raid* 缓存卡问题，最终确认了是有 *raid* 卡，但上面没有电池！
- 连夜购买 *raid* 卡电池，半夜上架
- 效果： *db io %util* 基本回落到 *10%*，偶尔跳到 *20%*，更关键的是 *await/svctm* 都降到 *0.x* 毫秒了
- 这时米聊同时在线四五万， *DB* 毫无压力 ^-^

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- 终于暂时没被业务推了，可以缓下来规划了
- 评估了 *FusionIO* ，当时没选：贵； *IO* 瓶颈已经不明显了
- 采购了新的专有 *DB* 服务器： *HP380G7*
- 开始推 *Raid10* ，分离 *data file* 和 *binlog* 到不同磁盘
- *mysql 5.1 innodb plugin*
- *innodb_flush_method = O_DIRECT*
- *innodb_file_per_table* ； *sync_binlog = 0*
- *Kernel io deadline* 调度算法；内存 *swappiness = 0*
- 重视 *dmesg*

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- 中间服务层引入读写分离的支持，离线 *job* 都走 *slave DB*
- 引入 *memcached*，挡掉大量在线业务读请求
- 开始根据 *slow query* 优化 *SQL* 和表结构
- 重视业务发布前的 *sql review*
- 开始权限分离，只有兼职 *DBA* 有 *os/mysql root* 权限
- *oom_adj* 给 *-17*，预防被 *oom killer* 干掉
- *pt-ioprofile* 是个好东西

一个“兼职 DBA” 的数据库运维经验谈

- 经过程序架构调整， *DB* 持续优化。现在经过近百倍增长后的业务（几十万同时在线，过千万注册用户），用户 + 关系系统还是一个主库就能抗住，并且余量还有若干倍， *io %util* 基本稳定在 *10~20%* 左右，业务 *SQL* 层面是读多于写 *(qps* 基本在 *1500~2000)* ，磁盘层面是写多于读
- *rd_avkb 5~6; rd_mb_s 0.3; rd_rt 4~7ms*
- *wr_avkb 20~70; wr_mb_s 2; wr_rt 0.1ms*
- 把随机小块读请求尽量优化掉：写不可避免，读对写的性能干扰影响很大； *cfq/deadline* 对读请求比较“偏心”；写不急，我们有 *raid* 缓存削峰加合并；

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- *Case: root* 超过连接限制连不上 *instance*， 接着 *mysql crash*
- 经验措施： *ulimit* 允许 *core*， 自己源码编译 *mysql*， 这样 *crash* 了好定位 *root cause*

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- *Case:* 某离线运营数据库 *load* 高
- 排查方法：先是确认 *cpu-bound*；接着 *oprofile* 一把看到是 *query cache* 相关代码排榜首
- 经验措施：根据当时我们业务的特征，线上 *query cache* 设为 *demand* 模式，这样高级程序员通过 *hint* 也可以利用这个特性，同时又避免类似事情发生

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- *Case:* 富媒体业务定时删过期数据引起的每日凌晨报警
- 调查：分析 *SQL* 和表结构，确认是过期数据量较大

  ---TRANSACTION 37633CA5, ACTIVE 1715 sec, process no 9628, OS thread id 1082059072 fetching rows

  mysql tables in use 1, locked 1

  178528 lock struct(s), heap size 15907256, 6044203 row lock(s), undo log entries 1166295
- 措施：通过 *oak-chunk-update* 逐个 *chunk* 删，每次 *1k* 条

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- *Case:* 某运营统计数据库直接被新来同学的脚本自动化 *drop* 掉，由于是非核心数据库，没有每日备份
- 措施 *1*：幸好一星期前某开发有个 *mysqldump* 导数据到线下开发，并且 *binlog* 都在，于是苦逼的恢复了一下午。。。
- 措施 *2*：不管啥线上 *DB*，都要每日备份
- 措施 *3*：权限最小化，只分配 *crud* 中的几种

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- *Case:* 内存扩容 *(16G->32/64G)*，调大 *bp* 后，凌晨监控说物理内存尚有余量不少的情形下，开吃 *swap* 了
- 调查：研究硬件，确认支持 *numa*
- 措施 *1*：先定时 *echo 1 > /proc/sys/vm/drop_caches*
- 措施 *2*：下次 *instance* 重启期间， *numactl –interleave all*

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- *Case: (* 兄弟部门 *)SQL* 性能不好
- 调查：仔细分析 *query log*，和实际的多次 *explain* 发现同一语句会走不同索引，说明执行计划乱跳
- 措施：直接 *force index* 解决，毕竟 *optimizer* 没有实际 *DBA* 或开发更了解实际数据分布情形

一个“兼职 DBA” 的数据库运维经验谈

- 一年内业务百倍增长 DB 遇到的“痛”
- *Case:* 整个数据库 *hung*
- 调查：因为已经没法用 *mysql root* 连进， *gcore* 会太 *heavy* ，当时直接 *pstack* 若干次后，强制重启实例了。通过分析 *stack trace* ，确认是 *mysql bug60682*
- 措施：将多余的各种监控系统停掉，降低 *show engine innodb status* 请求频率；重视 *kernel mutex* ； *DDL* 时尤其注意，不要因为执行时间过长就随意 *kill session*

一个“兼职 DBA” 的数据库运维经验谈

- 过去一年在小公司里的收获
- 尽量充分了解并发挥好硬件性能
- 做运维真是如履薄冰，心惊胆战，还是做开发舒服些
- 需要深入了解 *OS* 底层才能运维好上层应用
- 多跟进业界大厂做法，才能在时间、人才资源很紧缺的情况下更有效的做好 *DBA* 运维，少踩坑、不踩深坑大坑
- 靠谱的程序员、靠谱的业务架构会让 *DBA* 轻松加愉快
- 等到其它各方面都被榨的差不多了，再榨 *mysql internal source code* 不迟，毕竟这个路子一般讲需要专人专职投入，性价比不是足够高，尤其是 *1)* 创业公司资源紧张 *2)* 现在业界 *mysql* 代码测试这块还没完全跟上发展节奏

一个“兼职 DBA” 的数据库运维经验谈

说实话，我来这儿有一个原因是：

宣传下，希望小米能招到个靠谱的 *Mysql DBA*

求加入