MySQL运维那些事

叶金荣

2015.06.18

about me

- 叶金荣，网络常用ID：yejr
- 2006年创办国内首个MySQL专业技术网站 http://imysql.com
- 国内最早的MySQL推广者，资深MySQL专家，10余年MySQL经验，擅长MySQL性能优化、架构设计、故障排查
- MySQL经历
  - MySQl 3.32.48 ~ now(5.7)
  - 2000 ~ now
  - 2012, ORACLE ACE(MySQL)

agenda

- 模式设计、优化
- 关于架构设计
- 运维经验
- 几个案例分享
- 关于运行环境

模式设计、优化

InnoDB表主键设计

- 默认地，采用INT AUTO_INCREMENT作为主键
- 读多写少的从库或归档库除外
- 优点
  - B+Tree分裂代价小，写入效率相对高9%+
  - 该字段和业务无关，变更灵活
  - 类似分页场景中，表连接效率更高
- 缺点
  - 基于secondary key查找时，需要多一次读
  - 多占用一些存储空间

案例： http://t.cn/RhJlR9n http://t.cn/RPGkTeI

BIGINT/char(3)/TINYINT

- type列：8-bytes vs 3（实际*3）-bytes vs 1-byte
- 1000万行记录，type列分别采用上述三种类型
- 存储空间对比
  - char(3)相比tinyint大280MB
  - bigint相比tinyint大136MB

BIGINT/char(3)/TINYINT

- type列：8-bytes vs 3（实际*3）-bytes vs 1-byte
- 1000万行记录，type列分别采用上述三种类型
- 根据PK随机读取1000万次
  - char(3)相比tinyint多691MB
  - bigint相比tinyint多3758MB

BIGINT/char(3)/TINYINT

- type列：8-bytes vs 3（实际*3）-bytes vs 1-byte
- 1000万行记录，type列分别采用上述三种类型
- 根据type列（有索引）随机读取50万次
  - char(3)相比tinyint多423MB
  - bigint相比tinyint多446MB
  - char(3)还有隐式类型转换风险

- 综上，强烈建议小范围的枚举型采用tinyint

TEXT/BLOB有多糟糕

- 超长TEXT/BLOB字段off-page存储
  - I/O效率差，消耗磁盘空间大，读写以及搜索效率也会差很多
  - SELECT * 的时候也会完全读取大字段
  - 哪怕只多出来1个字节，也会独自占用一个page
- 案例
  - 大量的TEXT字段尽量对齐重整
  - 原先100G的大表拆分成多个子表
  - 总大小也只有原来的1/4
- 建议：少用TEXT，或者分离到子表上

还要用TIMESTAMP代替DATETIME吗

- 5.6.5以前确实强烈建议这么做
- 5.6.5以后基本可忽略这个规范了
- 5-bytes（之前是8-bytes） vs 4-bytes
- DATETIME支持的范围更大
- DATETIME也支持初始及自动更新成CURRENT_TIMESTAMP
- 综上，根据实际情况选择吧

联合索引怎么用

- 哪个不能完整用到联合索引k1（c1, c2, c3）

WHERE c1 = ? AND c2 IN (?, ?) AND c3 = ? √

WHERE c1 = ? AND c2 =? ORDER BY c3 √

WHERE c3 = ? AND c1 = ? AND c2 IN (?, ?) √

WHERE c1 = ? AND c2 IN (?, ?) ORDER BY c3 x

建议新增 (c1, c3) 索引

JOIN中驱动表的选择

```
[employees]>EXPLAIN SELECT b.emp_no,a.title,a.from_date,a.to_date
    -> FROM titles a
    -> INNER JOIN employees b
    -> on a.emp_no = b.emp_no;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	b	index	PRIMARY	PRIMARY	4	NULL	299246	Using index
1	SIMPLE	a	ref	PRIMARY,emp_no	PRIMARY	4	employees.b.emp_no	1	NULL

JOIN中驱动表的选择

```
[employees]>EXPLAIN SELECT b.emp_no,a.title,a.from_date,a.to_date
    -> FROM titles a
    -> LEFT JOIN employees b
    -> on a.emp_no = b.emp_no;
```

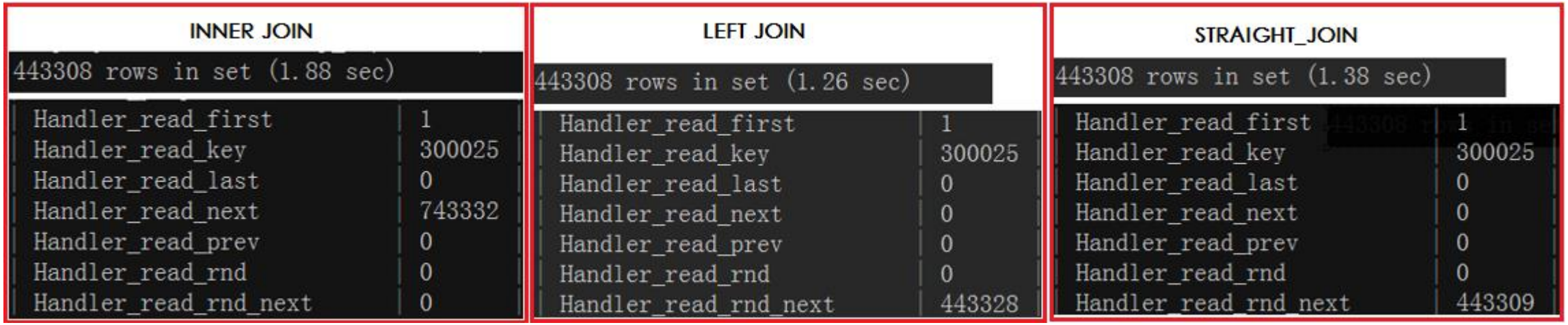
id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	a	ALL	NULL	NULL	NULL	NULL	441832	NULL
1	SIMPLE	b	eq_ref	PRIMARY	PRIMARY	4	employees.a.emp_no	1	Using index

JOIN中驱动表的选择

```
[employees]>EXPLAIN SELECT b.emp_no,a.title,a.from_date,a.to_date
    -> FROM titles a
    -> STRAIGHT_JOIN employees b
    -> on a.emp_no = b.emp_no;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	a	ALL	PRIMARY,emp_no	NULL	NULL	NULL	441832	NULL
1	SIMPLE	b	eq_ref	PRIMARY	PRIMARY	4	employees.a.emp_no	1	Using index

JOIN中驱动表的选择



INNER JOIN

443308 rows in set (1.88 sec)

Handler_read_first	1
Handler_read_key	300025
Handler_read_last	0
Handler_read_next	743332
Handler_read_prev	0
Handler_read_rnd	0
Handler_read_rnd_next	0

LEFT JOIN

443308 rows in set (1.26 sec)

Handler_read_first	1
Handler_read_key	300025
Handler_read_last	0
Handler_read_next	0
Handler_read_prev	0
Handler_read_rnd	0
Handler_read_rnd_next	443328

STRAIGHT_JOIN

443308 rows in set (1.38 sec)

Handler_read_first	1
Handler_read_key	300025
Handler_read_last	0
Handler_read_next	0
Handler_read_prev	0
Handler_read_rnd	0
Handler_read_rnd_next	443309

- INNER JOIN驱动顺序由优化器指定，但有时会选择错误，可用LEFT JOIN或STRAIGHT_JOIN指定顺序，不过要注意结果的正确性
- 案例：http://t.cn/RZV2Fya

SQL怎么写不会踩坑

- 写法1：UPDATE t1 SET c3 = ‘v3’ WHERE
  c1 = @c1 AND
  c2 = @c2;

- 写法2：UPDATE t1 SET c3 = ‘v3’ WHERE c1 = @c1
  AND c2 = @c2;

- 写法3：UPDATE t1 SET c3 = ‘v3’ WHERE 1 = 1
  AND c1 = @c1
  AND c2 = @c2;

模式设计总结

- 字段长度够用就好
- 用好索引，尤其是联合索引
- 关注新版本的变化，比如5.7里的很多新特性非常诱人
- 自己多动手，不要“听说、据说”

关于架构设计

分库分表真的有必要吗

- 为什么提倡分库分表
- 好处，单实例压力小，单节点故障影响范围小
- 坏处，架构更复杂，逻辑实现更麻烦
- 那么，你想清楚了吗
- 建议量力而行

前端cache/nosql层重要吗

- 如果没有cache/nosql层会怎样
- 数十万级tps vs 数千级tps
- 每秒数万次简单UPDATE COUNT+1就能搞垮MySQL
- 点击数、阅读数无需实时存盘，更新到nosql层
- 不常更新但频繁读取的数据，放在cache层
- 简单K-V数据放在nosql层

什么样的高可用方案是最合适的

- 简单的，用keepalived管理双主/主从两个节点
- 略复杂的，用MHA管理1主（或双主）多从
- 或者，也可以考虑PXC方案
- 更大一些的，用ZK管理集群
- 哪个用的最顺手，就选择哪个
- 全自动还是半自动切换呢？心里没底时，就先半自动吧

运维经验

单表大小建议

- 物理大小不超过10G
- 行数不过亿
- 平均物理行长度不超过8KB

最好还是关闭query cache吧

- query cache大多数情况下鸡肋，最好关闭
- QC锁是全局锁，每次更新QC的内存块锁代价高，很容易出现Waiting for query cache lock状态
- 想关闭query cache的话，size和type两个选项都设为0
- 参考：http://t.cn/RAF4d7z http://t.cn/RAF4d7Z

分支版本选择个人看法

- 优先选择Percona
- 其次才是MariaDB
- 未来更看好MariaDB，其更具核心竞争力
- 但MariaDB和ORACLE MySQL越离越远，兼容性是个大问题
- 用Percona/MariaDB时，记得开thread pool

几个案例分享

案例一

- too many connections的处理
- 常规的做法是这样：想办法杀掉多余的连接，加大连接数
- 其实应该是这样：限制连接数，设定max_user_connections
- 如果是这样呢：extra-port
- 建议：定时检查，干掉慢查询，避免阻塞，自我保护

案例二

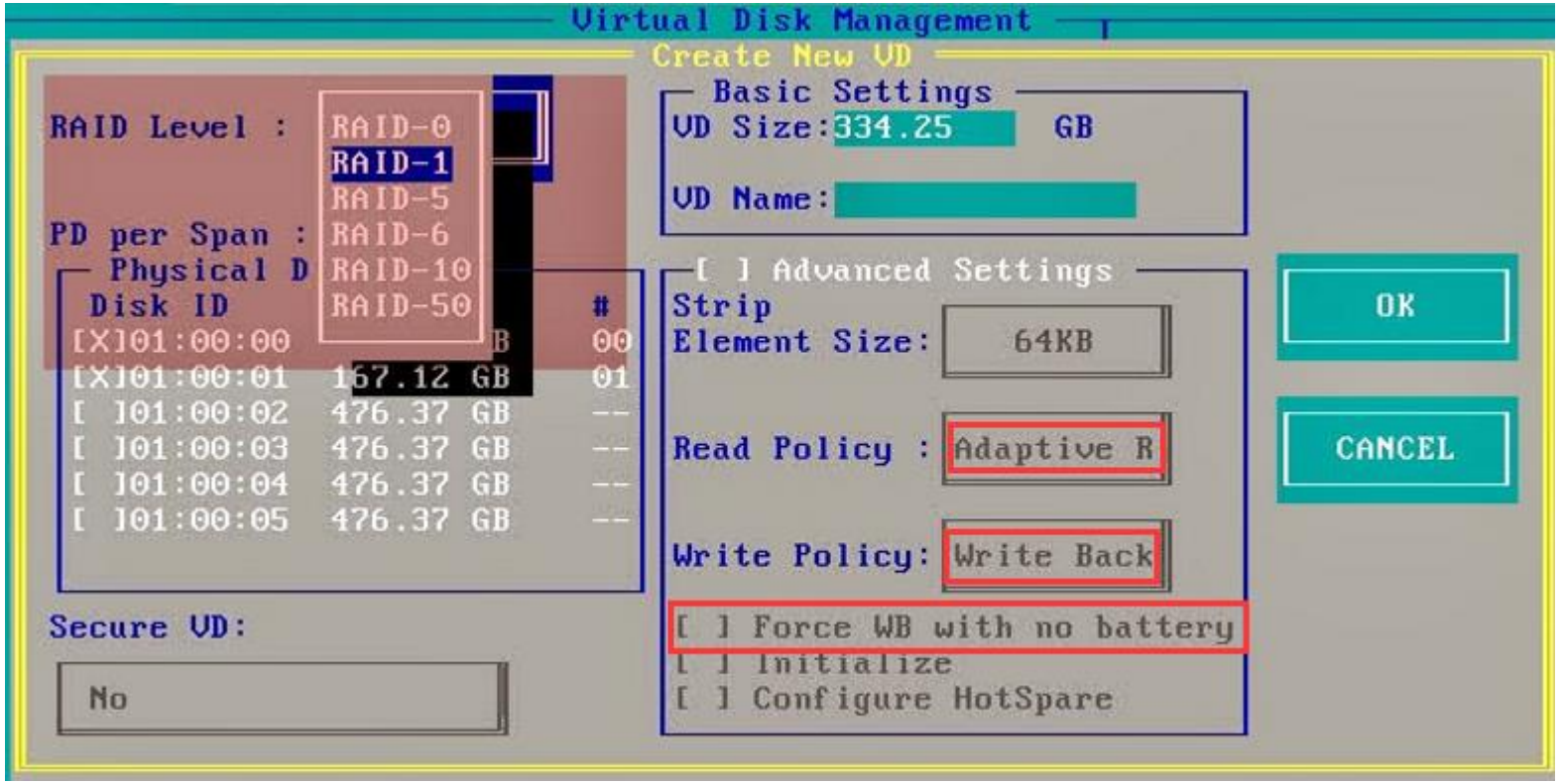
- 新初始化的slave实例启动时报告InnoDB数据页损坏错误
- 原因
  - redo log file大小和my.cnf配置值不一样
  - 5.6.8后会判断innodb_log_file_size值，决定是否自动重建redo log
  - 重建后，导致部分数据丢失，发生page crash

案例： http://t.cn/RAYTCbP

关于运行环境

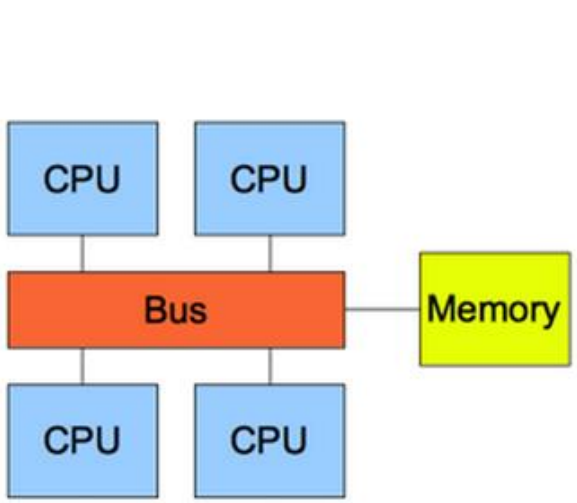
运行环境

- raid卡： FORCE WB

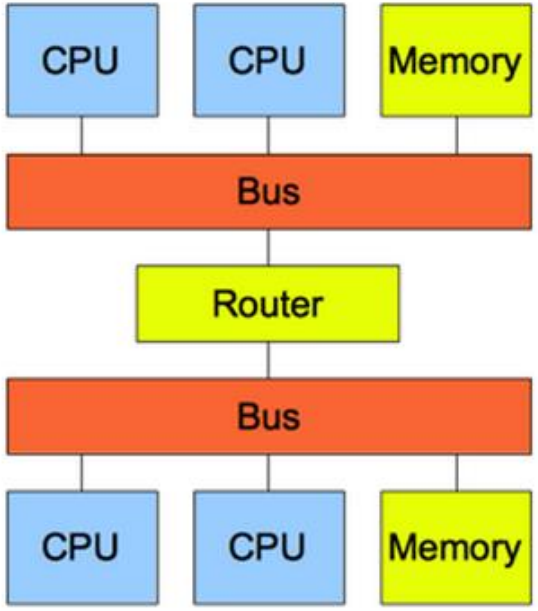


运行环境

- 关闭numa



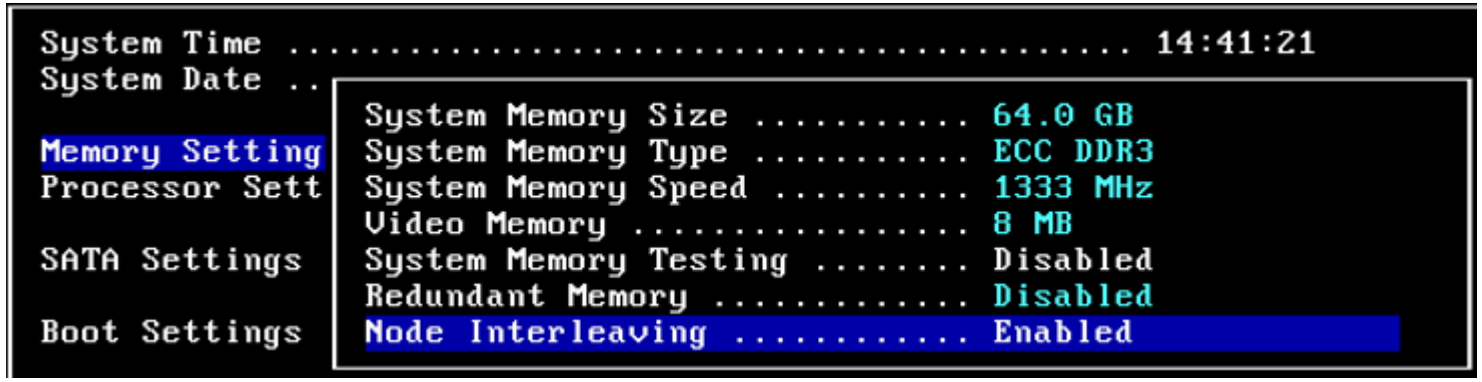
SMP Architecture



NUMA Architecture

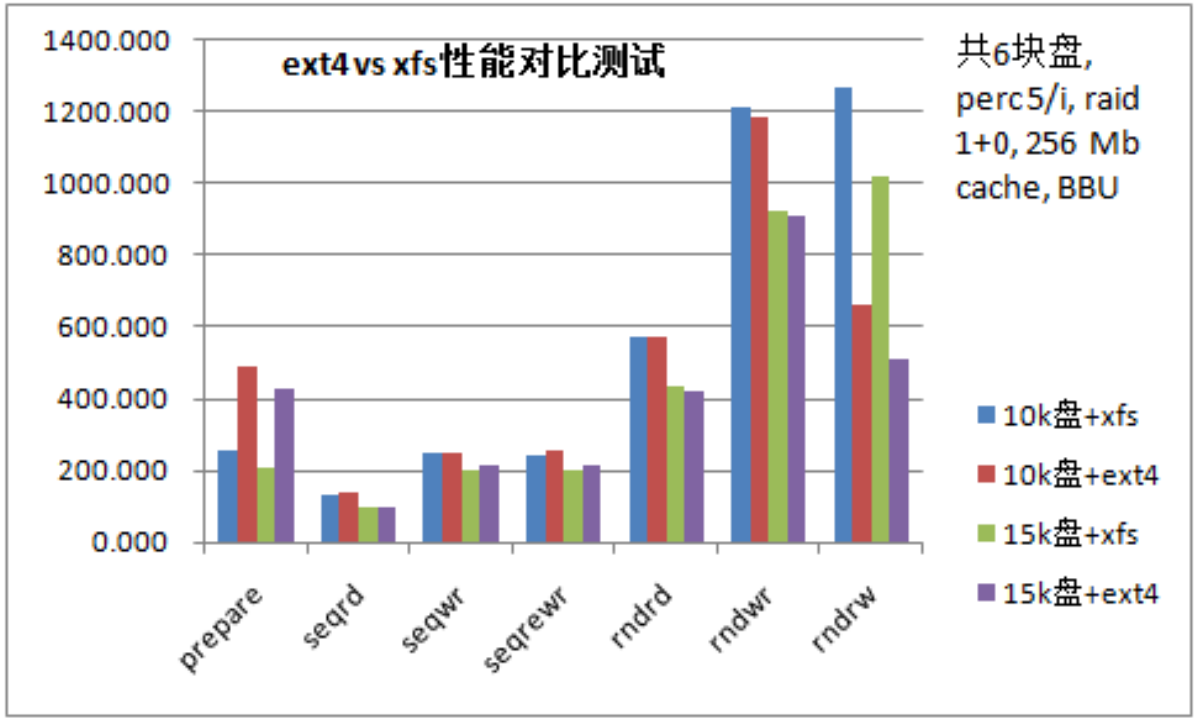
运行环境

- 关闭numa


System Time 14:41:21
System Date ..
Memory Setting
Processor Sett
SATA Settings
Boot Settings
System Memory Size ..... 64.0 GB
System Memory Type ..... ECC DDR3
System Memory Speed ..... 1333 MHz
Video Memory ..... 8 MB
System Memory Testing ..... Disabled
Redundant Memory ..... Disabled
Node Interleaving ..... Enabled

运行环境

- I/O，deadline
  - ext3早已渣的不行了
  - I/O压力不大时也可选择ext4，高I/O时选择xfs最可靠



	Directory contents	File allocation
xfs	B+ trees	B+ trees
ext4	Linked list, hashed B-tree	Extents/Bitmap

运行环境

- 内核相关参数
  - 内存、I/O相关
    - vm.swappiness，不高于5-10
    - /proc/sys/vm/dirty_background_ratio，不高于10
    - /proc/sys/vm/dirty_ratio，不高于30，比dirty_background_ratio大，避免I/O子系统hang住
    - /sys/block/xxx/queue/read_ahead_kb，对读为主的场景影响较大，其余可不关注
    - /sys/block/xxx/queue/nr_requests，对顺序写入为主的场景影响较大，其余可不关注
  - 网络相关
    - net.ipv4.tcp_tw_recycle = 1
    - net.ipv4.tcp_tw_reuse = 1

值得期待的5.7

- 官方号称比5.6快2倍多
  - 实际OLTP测试时，比Percona-5.6还要快8%，但高并发时则慢了10%
- 多源复制（multi-source replication）
  - 对于分库分表的场景尤其实用
- 在线扩展VARCHAR列长度
- 支持多个page cleaner线程
- 更好支持Fusion-io设备
- 在线修改innodb buffer pool大小
- EXPLAIN for Running Queries
- InnoDB新增只读事务模式
- InnoDB Faster & Parallel Flushing
- 以及查询优化器上的各种改进