MySQL 面试题

1， mysql 的复制原理以及流程。

（1） 先问基本原理流程，3 个线程以及之间的关联。

MySQL 的复制原理：Master 上面事务提交时会将该事务的 binlog event 写入 binlog file,然后 master 将 binlog event 传到 slave 上面，slave 应用该 binlog event 实现逻辑复制。

MySQL 的复制是基于如下 3 个线程的交互（多线程复制里面应该是 4 类线程）：

a. Master 上面的 binlog dump 线程，该线程负责将 master 的 binlog event 传到 slave；

b. Slave 上面的 IO 线程，该线程负责接收 Master 传过来的 binlog，并写入 relay log；

c. Slave 上面的 SQL 线程，该线程负责读取 relay log 并执行；

d. 如果是多线程复制，无论是 5.6 库级别的假多线程还是 MariaDB 或者 5.7 的真正的多线程复制，SQL 线程只做 coordinator，只负责把 relay log 中的 binlog 读出来然后交给 worker 线程，woker 线程负责具体 binlog event 的执行；

（2） 再问一致性延时性，数据恢复。

一致性可以从以下几个方面来讲：

a. 在 MySQL5.5 以及之前，slave 的 SQL 线程执行的 relay log 的位置只能保存在文件（relay-log.info）里面，并且该文件默认每执行 10000 次事务做一次同步到磁盘，这意味着 slave 意外 crash 重启时，SQL 线程执行到的位置和数据库的数据是不一致的，将导致复制报错，如果不重搭复制，则有可能会导致数据不一致。MySQL 5.6 引入参数 relay_log_info_repository，将该参数设置为 TABLE 时，MySQL 将 SQL 线程执行到的位置存到 mysql.slave_relay_log_info 表，这样更新该表的位置和 SQL 线程执行的用户事务绑定成一个事务，这样 slave 意外宕机后，slave 通过 innodb 的崩溃恢复可以把 SQL 线程执行到的位置和用户事务恢复到一致性的状态。

b. MySQL 5.6 引入 GTID 复制，每个 GTID 对应的事务在每个实例上面最多执行一次，这极大地提高了复制的数据一致性；

c. MySQL 5.5 引入半同步复制，用户安装半同步复制插件并且开启参数后，设置超时时间，可保证在超时时间内如果 binlog 不传到 slave 上面，那么用户提交事务时不会返回，直到超时后切成异步复制，但是如果切成异步之前用户线程提交时在 master 上面等待的时候，事务已经提交，该事务对 master

上面的其他 session 是可见的，如果这时 master 宕机，那么到 slave 上面该事务又不可见了，该问题直到 5.7 才解决；

d. MySQL 5.7 引入无损半同步复制，引入参 rpl_semi_sync_master_wait_point，该参数默认为 after_sync，指的是在切成半同步之前，事务不提交，而是接收到 slave 的 ACK 确认之后才提交该事务，从此，复制真正可以做到无损的了。

e. 可以再说一下 5.7 的无损复制情况下，master 意外宕机，重启后发现有 binlog 没传到 slave 上面，这部分 binlog 怎么办？？？分 2 种情况讨论，1 宕机时已经切成异步了，2 是宕机时还没切成异步？？？**这个怎么判断宕机时有没有切成异步呢？？？分别怎么处理？？？**

延时性：

可以讲下 5.5 是单线程复制，5.6 是多库复制（对于单库或者单表的并发操作是没用的），5.7 是真正意义的多线程复制，它的原理是基于 group commit，只要 master 上面的事务是 group commit 的，那 slave 上面也可以通过多个 worker 线程去并发执行。和 MairaDB10.0.0.5 引入多线程复制的原理基本一样。

数据恢复？？？他想问什么？？？

（3）再问各种工作遇到的复制 bug 的解决方法

复制的 bug？？？上面说的算吗？我还真没遇到过，5.6 的多库复制有时候自己会停止，我们写了一个脚本重新 start slave;待补充…

2， mysql 中 myisam 与 innodb 的区别，至少 5 点。

（1） 问 5 点不同

a. Innodb 支持事务，myisam 不支持；
b. Innodb 支持行级别锁，myisam 不支持；
c. Innodb 不能通过直接拷贝表文件的方法拷贝表到另外一台机器，myisam 支持；
d. Innodb 是索引组织表，myisam 是堆表；
e. Innodb 表不容易损坏，myisam 容易；
f. Innodb 表支持多种行格式，myisam 不支持；

（2） 问各种不同 mysql 版本的 2 者的改进

MySQL5.6 下 Innodb 引擎的主要改进：

（1） online DDL
（2） memcached NoSQL 接口
（3） transportable tablespace（alter table discard/import tablespace）

（4） MySQL 正常关闭时，可以 dump 出 buffer pool 的（space，page_no），重启时 reload，加快预热速度
（5） 索引和表的统计信息持久化到 mysql.innodb_table_stats 和 mysql.innodb_index_stats，可提供稳定的执行计划
（6） Compressed row format 支持压缩表

MySQL 5.7 innodb 引擎主要改进
（1） 修改 varchar 字段长度有时可以使用 online DDL
（2） Buffer pool 支持在线改变大小
（3） Buffer pool 支持导出部分比例
（4） 支持新建 innodb tablespace，并可以在其中创建多张表
（5） 磁盘临时表采用 innodb 存储，并且存储在 innodb temp tablespace 里面，以前是 myisam 存储
（6） 透明表空间压缩功能

Myisam 还要深入了解吗？？？我不了解它的各版本变化

（3） 2 者的索引的实现方式
都是 B+树索引，Innodb 是索引组织表，myisam 是堆表，索引组织表和堆表的区别要熟悉

3，问 mysql 中 varchar 与 char 的区别以及 varchar(50)中的 30 代表的涵义。

（1） varchar 与 char 的区别
在单字节字符集下，char（N）在内部存储的时候总是定长， 而且没有变长字段长度列表中。在多字节字符集下面，char(N)如果存储的字节数超过 N，那么 char（N）将和 varchar（N）没有区别。在多字节字符集下面，如果存储的字节数少于 N，那么存储 N 个字节，后面补空格，补到 N 字节长度。都存储变长的数据和变长字段长度列表。varchar(N)无论是什么字节字符集，都是变长的，即都存储变长数据和变长字段长度列表。

（2） varchar(50)中 50 的涵义
在早期 MySQL 版本中，50 代表字节数，现在代表字符数。

（3） int（20）中 20 的涵义
不影响内部存储，只是影响带 zerofill 定义的 int 时，前面补多少个 0，易于报表展示。

（4） 为什么 MySQL 这样设计？
不知道。。。

[备注] 本人也面试了近 12 个 2 年 mysql dba 经验的朋友，很少能回答出第（2）以及（4）题。

4， 问了 innodb 的事务与日志的实现方式。

（1） 有多少种日志

redo/undo

（2） 日志的存放形式

redo：在页修改的时候，先写到 redo log buffer 里面，然后写到 redo log 的文件系统缓存里面(fwrite)，然后再同步到磁盘文件（fsync）。

Undo：在 MySQL5.5 之前，undo 只能存放在 ibdata*文件里面，5.6 之后，可以通过设置 innodb_undo_tablespaces 参数把 undo log 存放在 ibdata*之外。

（3） 事务是如何通过日志来实现的，说得越深入越好。

基本流程如下：

因为事务在修改页时，要先记 undo，在记 undo 之前要记 undo 的 redo，然后修改数据页，再记数据页修改的 redo。Redo（里面包括 undo 的修改）一定要比数据页先持久化到磁盘。当事务需要回滚时，因为有 undo，可以把数据页回滚到前镜像的状态，崩溃恢复时，如果 redo log 中事务没有对应的 commit 记录，那么需要用 undo 把该事务的修改回滚到事务开始之前。如果有 commit 记录，就用 redo 前滚到该事务完成时并提交掉。

5， 问了 mysql binlog 的几种日志录入格式以及区别

（1） 各种日志格式的涵义

Statement/row 模式，mixed 模式是二者的结合

（2） 适用场景

在一条 SQL 操作了多行数据时，statement 更节省空间，row 更占用空间。但是 row 模式更可靠。

（3）结合第一个问题，每一种日志格式在复制中的优劣。

Statement 可能占用空间会相对小一些，传送到 slave 的时间可能也短，但是没有 row 模式的可靠。Row 模式在操作多行数据时更占用空间，但是可靠。

6， 问了下 mysql 数据库 cpu 飙升到 500%的话他怎么处理？

（1） 没有经验的，可以不问

（2）有经验的，问他们的处理思路

(下面只是我处理过的案例的思路，其他的思路需要大家补充)

当 cpu 飙升到 500%时，先用操作系统命令 top 命令观察是不是 mysqld 占用导致的，如果不是，找出占用高的进程，并进行相关处理。如果是 mysqld 造成的，show processlist，看

看里面跑的 session 情况，是不是有消耗资源的 sql 在运行。找出消耗高的 sql，看看执行计划是否准确，index 是否缺失，或者实在是数据量太大造成。一般来说，肯定要 kill 掉这些线程(同时观察 cpu 使用率是否下降)，等进行相应的调整(比如说加索引、改 sql、改内存参数)之后，再重新跑这些 SQL。也有可能是每个 sql 消耗资源并不多，但是突然之间，有大量的 session 连进来导致 cpu 飙升，这种情况就需要跟应用一起来分析为何连接数会激增，再做出相应的调整，比如说限制连接数等。

7， sql 优化。

（1）explain 出来的各种 item 的意义

（2）profile 的意义以及使用场景。

（3）explain 中的索引问题。

（1） explain 出来的各种 item 的意义

id:每个被独立执行的操作的标志，表示对象被操作的顺序。一般来说，id 值大，先被执行；如果 id 值相同，则顺序从上到下。

select_type：查询中每个 select 子句的类型。具体待明天补充。

table:名字，被操作的对象名称，通常的表名(或者别名)，但是也有其他格式。

partitions:匹配的分区信息。

type:join 类型。具体指待明天补充。

possible_keys：列出可能会用到的索引。

key:实际用到的索引。

key_len:用到的索引键的平均长度，单位为字节。

ref:表示本行被操作的对象的参照对象，可能是一个常量用 const 表示，也可能是其他表的 key 指向的对象，比如说驱动表的连接列。

rows:估计每次需要扫描的行数。

filtered:rows*filtered/100 表示该步骤最后得到的行数(估计值)。

extra:重要的补充信息。具体待明天补充。

（2） profile 的意义以及使用场景。

Profile 用来分析 sql 性能的消耗分布情况。当用 explain 无法解决慢 SQL 的时候，需要用 profile 来对 sql 进行更细致的分析，找出 sql 所花的时间大部分消耗在哪个部分，确认 sql 的性能瓶颈。(我用的也不多，期待更好的答案)

（3） explain 中的索引问题。

Explain 结果中，一般来说，要看到尽量用 index(type 为 const、ref 等，key 列有值)，避免使用全表扫描(type 显式为 ALL)。比如说有 where 条件且选择性不错的列，需要建立索引。被驱动表的连接列，也需要建立索引。被驱动表的连接列也可能会跟 where 条件列一起建立

联合索引。当有排序或者 group by 的需求时，也可以考虑建立索引来达到直接排序和汇总的需求。

8, 备份计划，mysqldump 以及 xtranbackup 的实现原理，

（1） 备份计划

（2）备份恢复时间

（3）备份恢复失败如何处理

原理：

mysqldump

mysqldump 属于逻辑备份。加入--single-transaction 选项可以进行一致性备份。后台进程会先设置 session 的事务隔离级别为 RR(SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ)，之后显式开启一个事务(START TRANSACTION /*!40100 WITH CONSISTENT SNAPSHOT */)，这样就保证了该事务里读到的数据都是事务事务时候的快照。之后再把表的数据读取出来。如果加上--master-data=1 的话，在刚开始的时候还会加一个数据库的读锁(FLUSH TABLES WITH READ LOCK),等开启事务后，再记录下数据库此时 binlog 的位置(show master status)，马上解锁，再读取表的数据。等所有的数据都已经导完，就可以结束事务。

Xtrabackup:

xtrabackup 属于物理备份，直接拷贝表空间文件，同时不断扫描产生的 redo 日志并保存下来。最后完成 innodb 的备份后，会做一个 flush engine logs 的操作(老版本在有 bug，在 5.6 上不做此操作会丢数据)，确保所有的 redo log 都已经落盘(涉及到事务的两阶段提交概念，因为 xtrabackup 并不拷贝 binlog，所以必须保证所有的 redo log 都落盘，否则可能会丢最后一组提交事务的数据)。这个时间点就是 innodb 完成备份的时间点，数据文件虽然不是一致性的,但是有这段时间的 redo 就可以让数据文件达到一致性(恢复的时候做的事情)。然后还需要 flush tables with read lock，把 myisam 等其他引擎的表给备份出来，备份完后解锁。这样就做到了完美的热备。

备份计划：

视库的大小来定，一般来说 100G 内的库，可以考虑使用 mysqldump 来做，因为 mysqldump 更加轻巧灵活，备份时间选在业务低峰期，可以每天进行都进行全量备份(mysqldump 备份出来的文件比较小，压缩之后更小)。

100G 以上的库，可以考虑用 xtranbackup 来做，备份速度明显要比 mysqldump 要快。一般是选择一周一个全备，其余每天进行增量备份，备份时间为业务低峰期。

备份恢复时间：

物理备份恢复快，逻辑备份恢复慢

备份恢复失败如何处理：

首先在恢复之前就应该做足准备工作,避免恢复的时候出错。比如说备份之后的有效性检查、权限检查、空间检查等。如果万一报错，再根据报错的提示来进行相应的调整。

9， 500 台 db，在最快时间之内重启。

可以使用批量 ssh 工具 pssh 来对需要重启的机器执行重启命令。也可以使用 salt（前提是客户端有安装 salt）或者 ansible（ansible 只需要 ssh 免登通了就行）等多线程工具同时操作多台服务器

10， 在当前的工作中，你碰到到的最大的 mysql db 问题是？

可以选择一个处理过的比较棘手的案例，或者选择一个老师在课程上讲过的死锁的案例;

没有及时 Purge + insert 唯一索引造成的死锁：具体案例可以参考学委笔记。

11， innodb 的读写参数优化

（1）读取参数，global buffer pool 以及 local buffer

（2）写入参数

（3）与 IO 相关的参数

（4）缓存参数以及缓存的适用场景

（1）读取参数，global buffer pool 以及 local buffer

Global buffer：

Innodb_buffer_pool_size

innodb_log_buffer_size

innodb_additional_mem_pool_size

local buffer(下面的都是 server 层的 session 变量，不是 innodb 的)：

Read_buffer_size

Join_buffer_size

Sort_buffer_size

Key_buffer_size

Binlog_cache_size

（2）写入参数

insert_buffer_size

innodb_double_write

innodb_write_io_thread

innodb_flush_method

（3）与 IO 相关的参数

Sync_binlog

Innodb_flush_log_at_trx_commit

Innodb_lru_scan_depth

Innodb_io_capacity

Innodb_io_capacity_max

innodb_log_buffer_size

innodb_max_dirty_pages_pct

（4）缓存参数以及缓存的适用场景

指的是查询缓存吗？？？使用于读多写少，如分析报表等等

query_cache_size

query_cache_type

query_cache_limit

maximumquery_cache_size

12 ，请简洁地描述下 MySQL 中 InnoDB 支持的四种事务隔离级别名称，以及逐级之间的区别？

四种隔离级别：

read uncommitted

read committed

repeatable read

serializable

不同级别的现象：

脏读：一个事务可以读取到另一个事务尚未提交的数据

不可重复读：两个事务读取同一条记录，两次读取的结果不一样

幻读：在一个事务中，由于其他插入操作事务的提交，导致返回了以前不存在的记录

不同的隔离级别有不同的现象，并有不同的锁定/并发机制，隔离级别越高，数据库的并发性就越差。

隔离级别	脏读	非重复读	幻读
read uncommitted	允许		
read committed		允许	允许
repeatable read			允许(MySQL 不允许)
serializable			

13，表中有大字段 X（例如：text 类型），且字段 X 不会经常更新，以读为为主，请问

（1）您 是选择拆成子表，还是继续放一起？

（2）写出您这样选择的理由？

我也不清楚是放一起还是拆开来。。。

14，MySQL 中 InnoDB 引擎的行锁是通过加在什么上完成（或称实现）的？为什么是这样子的？

InnoDB 存储引擎的锁是通过加在索引上面完成的，如果表没有创建索引，InnoDB 会自动创建一个 6 字节的自增索引。

因为 Innodb 是索引组织表，通过索引去找对应的数据行。

1　2 年 MySQL DBA 经验

其中许多有水分，一看到简历自我介绍，说公司项目的时候，会写上 linux 系统维护，mssql server 项目，或者 oracle data gard 项目，一般如果有这些的话，工作在 3 年到 4 年的话，他的 2 年 MySQL DBA 管理经验，是有很大的水分的。刚开始我跟领导说，这些不用去面试了，肯定 mysql dba 经验不足，领导说先面面看看，于是我就面了，结果很多人卡在基础知识这一环节之上，比如：

（1）有的卡在复制原理之上

（2）有的卡在 binlog 的日志格式的种类和分别

（3）有的卡在 innodb 事务与日志的实现上。

（4）有的卡在 innodb 与 myisam 的索引实现方式的理解上面。

.....

关于基础知识考查点，请参考我整理的基础面试点总结：

http://blog.csdn.net/mchdba/article/details/13505701

个人觉得如果有过真正的 2 年 mysql 专职 dba 经验，那么肯定会在 mysql 的基本原理上有所研究，因为很多问题都不得不让你去仔细研究各种细节，而自 己研究过的细节肯定会记忆深刻，别人问起一定会说的头头是道，起码一些最基本的关键参数比如 Seconds_Behind_Master 为 60 这个值 60 的准确涵义，面试了 10+的 mysql dba，没有一个说的准确，有的说不知道忘记了，有的说是差了 60 秒，有的说是与主上执行时间延后了 60 秒。

上面的内容基本都在上面的问题中，但是那个 seconds_behind_master 的值具体是怎么计算的？在源码里面的计算公式是什么？这里我们还没完全确定。。。

2 对于简历中写有熟悉 mysql 高可用方案

我一般先问他现在管理的数据库架构是什么，如果他只说出了主从，而没有说任何 ha 的方案，那么我就可以判断出他没有实际的 ha 经验。不过这时候也不能就是 断定他不懂 mysql 高可用，也许是没有实际机会去使用，那么我就要问 mmm 以及 mha 以及 mm+keepalived 等的原理实现方式以及它们之间的优 势和不足了，一般这种情况下，能说出这个的基本没有。

mmm 那东西好像不靠谱，据说不稳定，但是有人在用的，我只在虚拟机上面用过，和 mysql-router 比较像，都是指定可写的机器和只读机器。MHA 的话一句话说不完，可以翻翻学委的笔记。

3 对于简历中写有批量 MySQL 数据库服务器的管理经验

这个如果他说有的话，我会先问他们现在实际线上的 mysql 数据库数量有多少，分多少个节点组，最后问这些节点组上面的 slow log 是如何组合在一起来统计分析的。如果这些他都答对了，那么我还有一问，就是现在手上有 600 台数据库，新来的机器，Mysql 都安装好了，那么你如 何在最快的时间里面把这 600 台 mysql 数据库的 mysqld 服务启动

起来。这个重点在于最快的时间，而能准确回答出清晰思路的只有 2 个人。

slow log 分析：可以通过一个管理服务器定时去各台 MySQL 服务器上面 mv 并且 cp slow log，然后分析入库，页面展示。

最快的时间里面启动 600 台服务器：肯定是多线程。可以用 pssh，ansible 等多线程批量管理服务器的工具

4　对于有丰富的 SQL 优化的经验

首先问 mysql 中 sql 优化的思路，如果能准备说出来，ok，那么我就开始问 explain 的各种参数了，重点是 select_type，type，possible_key, ref,rows,extra 等参数的各种值的含义，如果他都回答正确了，那么我再问 file sort 的含义以及什么时候会出现这个分析结果，如果这里他也回答对了，那么我就准备问 profile 分析了，如果这里他也答对了，那么我就会再问一个问 题，那是曾经 tx 问我的让我郁闷不已的问题，一个 6 亿的表 a，一个 3 亿的表 b，通过外间 tid 关联，你如何最快的查询出满足条件的第 50000 到第 50200 中的这 200 条数据记录。

Explain 在上面的题目中有了，这里就不说了。

如何最快的查询出满足条件的第 50000 到第 50200 中的这 200 条数据记录？这个我想不出来！

关于 explain 的各种参数，请参考：http://blog.csdn.net/mchdba/article/details/9190771

5　对于有丰富的数据库设计经验

这个对于数据库设计我真的没有太多的经验，我也就只能问问最基础的，mysql 中 varchar(60) 60 是啥含义，int(30)中 30 是啥含义？ 如果他都回答对了，那么我就问 mysql 中为什么要这么设计呢？ 如果他还回答对了，我就继续问 int(20)存储的数字的上限和下限是多少？这个问题难道了全部的 mysql dba 的应聘者，不得不佩服提出这个问题的金总的睿智啊，因为这个问题回答正确了，那么他确实认认真真地研究了 mysql 的设计中关于字段类型的细节。至 于丰富的设计数据库的经验，不用着急，这不我上面还有更加厉害的 dba 吗，他会搞明白的，那就跟我无关了。

varchar(60)的 60 表示最多可以存储 60 个字符。int(30)的 30 表示客户端显示这个字段的宽度。为何这么设计？说不清楚，请大家补充。 int(20) 的上限为 2147483647(signed) 或者 4294967295(unsigned)。

6 关于 mysql 参数优化的经验。

首先问他它们线上 mysql 数据库是怎么安装的，如果说是 rpm 安装的，那么我就直接问调优参数了，如果是源码安装的，那么我就要问编译中的一些参数了，比如 my.cnf 以及存储引擎以及字符类型等等。然后从以下几个方面问起：

（1）mysql 有哪些 global 内存参数，有哪些 local 内存参数。

（2）mysql 的写入参数需要调整哪些？重要的几个写参数的几个值得含义以及适用场景，比如 innodb_flush_log_at_trx_commit 等。

（3）读取的话，那几个全局的 pool 的值的设置，以及几个 local 的 buffer 的设置。

（4）还有就是著名的 query cache 了，以及 query cache 的适用场景了，这里有一个陷阱，就是高并发的情况下，比如双十一的时候，query cache 开还是不开，开了怎么保证高并发，不开又有何别的考虑？

（1） mysql 有哪些 global 内存参数，有哪些 local 内存参数。

Global:

innodb_buffer_pool_size/innodb_additional_mem_pool_size/innodb_log_buffer_size/key_buffer_size/query_cache_size/table_open_cache/table_definition_cache/thread_cache_size

Local:

read_buffer_size/read_rnd_buffer_size/sort_buffer_size/join_buffer_size/binlog_cache_size/tmp_table_size/thread_stack/bulk_insert_buffer_size

（2） mysql 的写入参数需要调整哪些？重要的几个写参数的几个值得含义以及适用场景，比如 innodb_flush_log_at_trx_commit 等。(求补充)

sync_binlog 设置为 1，保证 binlog 的安全性。

innodb_flush_log_at_trx_commit：

0：事务提交时不将 redo log buffer 写入磁盘(仅每秒进行 master thread 刷新，安全性最差，性能最好)

1：事务提交时将 redo log buffer 写入磁盘(安全性最好，性能最差，**推荐生产使用**)

2：事务提交时仅将 redo log buffer 写入操作系统缓存(安全性和性能都居中，当 mysql 宕机但是操作系统不宕机则不丢数据，如果操作系统宕机，最多丢一秒数据)

innodb_io_capacity/innodb_io_capacity_max：看磁盘的性能来定。如果是 HDD 可以设置为 200-几百不等。如果是 SSD，推荐为 4000 左右。innodb_io_capacity_max 更大一些。

innodb_flush_method 设置为 O_DIRECT。

（3） 读取的话，那几个全局的 pool 的值的设置，以及几个 local 的 buffer 的设置。

Global:

innodb_buffer_pool_size:设置为可用内存的 50%-60%左右，如果不够，再慢慢上调。

innodb_additional_mem_pool_size：采用默认值 8M 即可。

innodb_log_buffer_size:默认值 8M 即可。

key_buffer_size:myisam 表需要的 buffer size，选择基本都用 innodb，所以采用默认的 8M 即可。

Local:
join_buffer_size：当 sql 有 BNL 和 BKA 的时候，需要用的 buffer_size(plain index scans，range index scans 的时候可能也会用到)。默认为 256k，建议设置为 16M-32M。
read_rnd_buffer_size：当使用 mrr 时，用到的 buffer。默认为 256k，建议设置为 16-32M。
read_buffer_size:当顺序扫描一个 myisam 表，需要用到这个 buffer。或者用来决定 memory table 的大小。或者所有的 engine 类型做如下操作:order by 的时候用 temporary file、SELECT INTO … OUTFILE 'filename' 、For caching results of nested queries。默认为 128K，建议为 16M。
sort_buffer_size：sql 语句用来进行 sort 操作(order by,group by)的 buffer。如果 buffer 不够，则需要建立 temporary file。如果在 show global status 中发现有大量的 Sort_merge_passes 值，则需要考虑调大 sort_buffer_size。默认为 256k，建议设置为 16-32M。
binlog_cache_size:表示每个 session 中存放 transaction 的 binlog 的 cache size。默认 32K。一般使用默认值即可。如果有大事务，可以考虑调大。
thread_stack：每个进程都需要有，默认为 256K，使用默认值即可。

（4） 还有就是著名的 query cache 了，以及 query cache 的适用场景了，这里有一个陷阱，就是高并发的情况下，比如双十一的时候，query cache 开还是不开，开了怎么保证高并发，不开又有何别的考虑？

建议关闭，上了性能反而更差。

7 关于事物隔离级别。
这个最起码你得知道那 4 个隔离级别的名字吧，以及之间的区别，还有你当前自己数据库里面设置的是哪个级别吧，可惜的是面试了那么多，能准备想都不用想的回答出的就只有一个，而且他只说出了前 3 个级别的名字。
关于事务隔离级别简单资料，请参考：http://blog.csdn.net/mchdba/article/details/12837427

Read Uncommitted:脏读，可以读取其他 session 未提交的脏数据。
Read Committed:读提交。允许不可重复读取，但不允许脏读取。提交后，其他会话可以看到提交的数据。
Repeatable Read: 禁止不可重复读取和脏读取、以及幻读(innodb 独有)。
Serializable: 事务只能一个接着一个地执行，但不能并发执行。事务隔离级别最高。

8 关于熟悉 mysql 的锁机制。
那么 gap 锁，next-key 锁，以及 innodb 的行锁是怎么实现的，以及 myisam 的锁是怎么实现的等，可惜能准备说出这些的只有一个 mysql dba。

Innodb 的锁的策略为 next-key 锁，即 record lock+gap lock。是通过在 index 上加 lock 实现的，如果 index 为 unique index，则降级为 record lock,如果是普通 index，则为 next-key lock，如果没有 index，则直接锁住全表。myisam 直接使用全表扫描。

9 关于熟悉 mysql 集群的。
我就问了 ndbd 的节点的启动先后顺序，再问配置参数中的内存配置几个重要的参数，再问 sql 节点中执行一个 join 表的 select 语句的实现流程是怎么走的？ok，能回答的也只有一个。
关于 mysql 集群入门资料，请参考：http://write.blog.csdn.net/postlist/1583151/all

答：mysql ndb 集群这个没接触过，期望有好的回答

10 关于有丰富的备份经验的
就问 mysqldump 中备份出来的 sql，如果我想 sql 文件中，一行只有一个 insert .... value() 的话，怎么办？如果备份需要带上 master 的复制点信息怎么办？或者 xtrabackup 中如何做到实时在线备份的？以及 xtrabackup 是如何做到带上 master 的复制点的信息的？ 当前 xtrabackup 做增量备份的时候有何缺陷？

能全部回答出来的没有一个，不过没有关系，只要回答出 mysqldump 或者 xtrabackup 其中一个的也可以。

1). --skip-extended-insert

2). --master-date=1

3). 因为 xtrabackup 是多线程，一个线程不停地在拷贝新产生的 redo 文件，另外的线程去备份数据库，当所有表空间备份完成的时候，它会执行 flush table with read lock 操作锁住所有表，然后执行 show master status; 接着执行 flush engine logs; 最后解锁表。执行 show master status; 时就能获取到 mster 的复制点信息，执行 flush engine logs 强制把 redo 文件刷新到磁盘。

4). xtrabackup 增量备份的缺陷不了解，在线上用 xtrabackup 备份没有发现什么缺陷。

11 关于有丰富的线上恢复经验的
就问你现在线上数据量有多大，如果是 100G，你用 mysqldump 出来要多久，然后 mysql 进去又要多久，如果互联网不允许延时的话，你又怎么做到 恢复单张表的时候保证 nagios 不报警。如果有人说 mysqldump 出来 1 个小时就 ok 了，那么我就要问问他 db 服务器是啥配置了，如果他说 mysql 进去 50 分钟搞定了，那么我也要问问他 db 机器啥配置了，如果是普通的吊丝 pc server，那么真实性，大家懂得。
然后如果你用 xtrabackup 备份要多久，恢复要多久，大家都知道 copy-back 这一步要很

久，那么你有没有办法对这一块优化。

答：线上恢复经验欠缺，不好回答，希望有这块经验的人可以补充下。

上一篇：Oracle 优化器(RBO 与 CBO)
下一篇：MySQL 面试题
ITPUB 论坛 | chinaunix 博客 | chinaunix 论坛