

面试案例

案例一：Mysql生产上线前标准化规划的准备工作

1.标准化规划

a.服务器硬件的选择

- 符合3-5年的硬件扩展。（从品牌、CPU、MEM、IO设备、网络设备、存储设备等各个层次进行合理建议。）

b.操作系统,

- 架构服务器系统与内核版本一致。（采用稳定系统Centos7.2以上双数版）

c.操作系统的硬件参数

- 关闭NUMA(三级别关闭 bios级别--->os grub级别--->数据库级别)
- 开启cpu高性能模式
- 阵列卡RAID配置 (raid10)
- 关闭THP
- 网卡绑定 (bonding技术)
- 存储多路径

d.操作系统层面的参数优化

- 文件句柄数nofile和进程数
- 防护墙与selinux
- 文件系统类型 (xfs)
- 不使用lvm
- io调度 (SAS : deadline SSD&PCI-E: noop)

e.数据库软件层面的标准化

- 数据库分支的选择 (oracle旗下的mysql)

f. 数据稳定版本的选择 (8.0.20)

- 没有大量bug修复的GA稳定版本 (双数) , 兼容性开发环境。

g. 数据库插件的选择

- 官方工具: workbench mysql-shell
- 第三方工具: PXB (备份) , PT , binlog2sql,FIO,sysbench(业务压测),stress, navicat/sqlyong(sql开发)

2.进行一周的烤机压测 (TOP检测)

a.CPU:使用stress工具进行一个监测cpu是否能够持久化的稳定 stress -c (cpu个数) -t (压测时间)

b.MEM:使用stress工具进行压测内存 stress -m (并发次数) --vm-bytes 1000M

stress工具混合压测 stress -c -m -d (压测线程个数)

c.FIO 进行定制化IO烤机压测 (顺序读写, 随机读写, 混合读写, 定制读写比例)

案例二：inplace本地升级Mysql5.6(5.6.46)版本升级到5.7(5.7.30)版本

0. 准备回退方案, 备份原数据库数据。

1. 安装新版本软件

2. 以优雅的方式关闭原数据库 (挂维护页)

3. 使用新版本软件“挂”旧版本数据启动(--skip-grant-tables ,--skip-networking)

4. 升级 :只是升级系统表。升级时间和数据量无关的。

5. 正常重启数据库。

6. 验证各项功能是否正常。(大型企业有专业人员测试)

7. 业务恢复。

案例三：批量更换业务库的存储引擎

项目背景：公司某库存在宕机后部分数据丢失的情况，业务压力大时服务非常卡。

问题分析：有可能是MySQL升级之后的遗留问题

1. 确认数据库版本

```
select version();
```

2. 确认业务表的存储引擎

```
select table_schema,table_name,engine from information_schema.tables
```

```
where table_schema not in ('mysql','information_schema');
```

最后发现有部分业务表的存储引擎为myisam引擎，MyISAM存储引擎只有表级锁，因此在高并发时，会有很高锁等待，并且其不支持事务，在断电时，有可能丢失数据。

解决方案：

对myisam引擎的业务表进行批量替换

1.将所有非InnoDB表查出来.

```
select table_schema,table_name,engine from information_schema.tables
where table_schema not in ('mysql','information_schema') and engine !='innodb';
```

2.生成批量替换的语句

```
SELECT CONCAT("ALTER TABLE ",table_schema,".",table_name,"
ENGINE=INNODB;")
FROM information_schema.tables
WHERE table_schema
NOT IN ('mysql','information_schema','performance_schema','sys')
AND ENGINE !='innodb'
INTO OUTFILE '/tmp/alter.sql';
```

案列四：配合数据库开发部门进行schema设计。

案列五（重点） 如何查看分析锁等待

1.查看默认锁等待实际

Bash | Copy

```
1  mysql> show variables like '%wait%';
2  +-----+-----+
3  | Variable_name | Value |
4  +-----+-----+
5  | innodb_lock_wait_timeout | 50 |
6
```

2.有关于锁等待信息的表在sys系统库下innodb_lock_waits

```

1  mysql> select * from innodb_lock_waits\G;
2  ***** 1. row *****
3      wait_started: 2021-04-17 21:17:05
4      wait_age: 00:00:48
5      wait_age_secs: 48
6      locked_table: `world`.`cry`
7      locked_table_schema: world
8      locked_table_name: cry
9      locked_table_partition: NULL
10     locked_table_subpartition: NULL
11     locked_index: PRIMARY
12     locked_type: RECORD
13     waiting_trx_id: 12832
14     waiting_trx_started: 2021-04-17 21:16:30
15     waiting_trx_age: 00:01:23
16     waiting_trx_rows_locked: 1
17     waiting_trx_rows_modified: 0
18     waiting_pid: 12
19     waiting_query: update cry set name='jj' where id=1
20     waiting_lock_id: 139751497162104:39:4:2:139751390782624
21     waiting_lock_mode: X,REC_NOT_GAP
22     blocking_trx_id: 12831
23     blocking_pid: 8
24     blocking_query: NULL
25     blocking_lock_id: 139751497162952:39:4:2:139751390788784
26     blocking_lock_mode: X,REC_NOT_GAP
27     blocking_trx_started: 2021-04-17 21:15:10
28     blocking_trx_age: 00:02:43
29     blocking_trx_rows_locked: 1
30     blocking_trx_rows_modified: 1
31     sql_kill_blocking_query: KILL QUERY 8
32     sql_kill_blocking_connection: KILL 8
33  1 row in set (0.00 sec)

```

3.分析锁等待（连接线程----> sql线程）

```

1  连接线程----> sql线程
2  1.首先
3  mysql> select * from innodb_lock_waits\G;
4  找出被阻塞和阻塞者的连接线程 (show processlist) 是谁?
5  ***** 1. row *****
6      wait_started: 2021-04-17 21:17:05
7      wait_age: 00:00:48
8      wait_age_secs: 48
9      locked_table: `world`.`cry` 查看
10     locked_table_schema: world
11     locked_table_name: cry
12     locked_table_partition: NULL
13     locked_table_subpartition: NULL
14     locked_index: PRIMARY  innodb
15     locked_type: RECORD  加锁
16     waiting_trx_id: 12832
17     waiting_trx_started: 2021-04-17 21:16:30
18     waiting_trx_age: 00:01:23
19     waiting_trx_rows_locked: 1  加锁
20     waiting_trx_rows_modified: 0
21     waiting_pid: 12  谁在
22     waiting_query: update cry set name='jj' where id=1  锁等
23     waiting_lock_id: 139751497162104:39:4:2:139751390782624
24     waiting_lock_mode: X,REC_NOT_GAP
25     blocking_trx_id: 12831
26     blocking_pid: 8  阻塞
27     blocking_query: NULL
28     blocking_lock_id: 139751497162952:39:4:2:139751390788784
29     blocking_lock_mode: X,REC_NOT_GAP
30     blocking_trx_started: 2021-04-17 21:15:10
31     blocking_trx_age: 00:02:43
32     blocking_trx_rows_locked: 1
33     blocking_trx_rows_modified: 1
34     sql_kill_blocking_query: KILL QUERY 8
35     sql_kill_blocking_connection: KILL 8
36
37  提取信息被阻塞的是12 (连接线程id号), 发起阻塞的是8 (连接线程id号)
38
39  2.通过performance_schema下的threads找到连接线程和sql线程的对应关系
40  select * from performance.threads\G;
41      THREAD_ID: 52
42      NAME: thread/sql/one_connection
43      TYPE: FOREGROUND
44      PROCESSLIST_ID: 12
45      PROCESSLIST_USER: root
46      PROCESSLIST_HOST: localhost
47      PROCESSLIST_DB: world
48      PROCESSLIST_COMMAND: Sleep
49      PROCESSLIST_TIME: 585
50      PROCESSLIST_STATE: NULL
51      PROCESSLIST_INFO: update cry set name='jj' where id=1  显示会话做
52      PARENT_THREAD_ID: NULL
53      ROLE: NULL
54      INSTRUMENTED: YES
55      HISTORY: YES
56      CONNECTION_TYPE: Socket
57      THREAD_OS_ID: 3863
58      RESOURCE_GROUP: USR_default
59
60  提取信息: 找到THREAD_ID: 52 (sql线程id号)
61
62  3.通过查询 performance_schema .events_statements_history 中对应的线程id号,
63  select * from performance_schema.events_statements_history where thread_id=52
64

```

案例六：mysql5.7 xtrabackup/mysqldump备份时数据库出现夯住的状态，所有修改查询都不能进行，只能查看

▼

Bash | Copy

一. 模拟操作

```
121     PROCESSLIST_HOST: localhost
122     PROCESSLIST_DB: world
123     PROCESSLIST_COMMAND: Query
124     PROCESSLIST_TIME: 4274
125     PROCESSLIST_STATE: User sleep
126     PROCESSLIST_INFO: select id,sleep(100) from city where id<100
127     PARENT_THREAD_ID: NULL
128     ROLE: NULL
129     INSTRUMENTED: YES
130     HISTORY: YES
131     CONNECTION_TYPE: Socket
132     THREAD_OS_ID: 1509
133     RESOURCE_GROUP: USR_default
134 5.2 找到对应的执行用户，与人进行沟通，了解情况，判断是否能够终止操作，释放锁。
135 为了保证正常语句可以执行，必须选择舍弃，在这里大事务停止回滚也会造成锁问题，可能
136
137
138 5.3 终止备份操作
139 mysql> select * from threads where THREAD_ID=59 \G;
140 ***** 1. row *****
141     THREAD_ID: 59
142     NAME: thread/sql/one_connection
143     TYPE: FOREGROUND
144     PROCESSLIST_ID: 18
145     PROCESSLIST_USER: root
146     PROCESSLIST_HOST: localhost
147     PROCESSLIST_DB: world
148     PROCESSLIST_COMMAND: Query
149     PROCESSLIST_TIME: 4540
150     PROCESSLIST_STATE: Waiting for global read lock
151     PROCESSLIST_INFO: flush tables with read lock
152     PARENT_THREAD_ID: NULL
```

```
153             ROLE: NULL
154             INSTRUMENTED: YES
155             HISTORY: YES
156             CONNECTION_TYPE: Socket
157             THREAD_OS_ID: 4030
158             RESOURCE_GROUP: USR_default
159
160 5.4 将对应的processlist_id kill掉
161 kill 18
162
163 6. 结论：我们备份操作要选择在业务低谷时期完成。
164
165 - 备份方法
```

5%E5%B9%B4%E7%9A%84%E7%A1%AC%E4%BB%B6%E6%89%A9%E5%B1%95%E3%80%82%EF%BC%88%E4%BB%8E%E5%93%81%E7'