数据库上云经典案例分析

玄惭

阿里云技术专家

自我介绍

玄惭

花名出自《天龙八部》

2012年加入阿里云RDS

负责RDS线上的稳定

历年RDS双11的负责人

目前负责RDS专家服务

目录 content

扫码观看大会视频

背景介绍

1、 某客户正在将本地的业务系统迁移上云

2、 在RDS上运行时间明显要比线下自建数据库运行时间要慢 1 倍

3、 导致客户系统割接延期的风险

经验分析

1、 数据库跨平台迁移（PG->MySQL、ORALCE->MySQL）

2、 跨版本升级(MySQL: 5.1->5.5、5.5->5.6)

3、 执行计划，优化器，参数配置，硬件配置

4、 网络延迟（跨可用区访问，公网延迟，网卡跑满）

优化器版本

- OPTIMIZER_SWITCH:
- index_merge=on,index_merge_union=on,index_merge_sort_union=on,
- index_merge_intersection=on,engine_condition_pushdown=on,
- index_condition_pushdown=on,mrr=on,mrr_cost_based=on,
- block_nested_loop=on,batched_key_access=off,materialization=on,
- semijoin=on,loosescan=on,firstmatch=on,
- subquery_materialization_cost_based=on,use_index_extensions=on

用户5.6.25，RDS的版本5.6.1.xx

SQL执行计划

key_len	ref	rows	Extra
NULL	NULL	39900	Using where
5	v.ID	1	Using where
5	v.ID	1	Using where
768	NULL	140	Using index
5	wdw.s.CID	285	Using where
4	wdw.ub.UID_CUS	1	NULL
5	wdw.ub.UID	1	Using index condition
5	wdw.up.ID	1	Using where
4	wdw.bg.BID	1	Using where
4	wdw.ub.UID	1	NULL
4	wdw.ub.UID	1	NULL
4	wdw.B.UID	1	Using where

rows=39900*1*1*140*285*1*1*1*1*1*1*1

参数配置

用户配置：

join_buffer_size = 128M

read_rnd_buffer_size = 128M

tmp_table_size = 128M

RDS配置

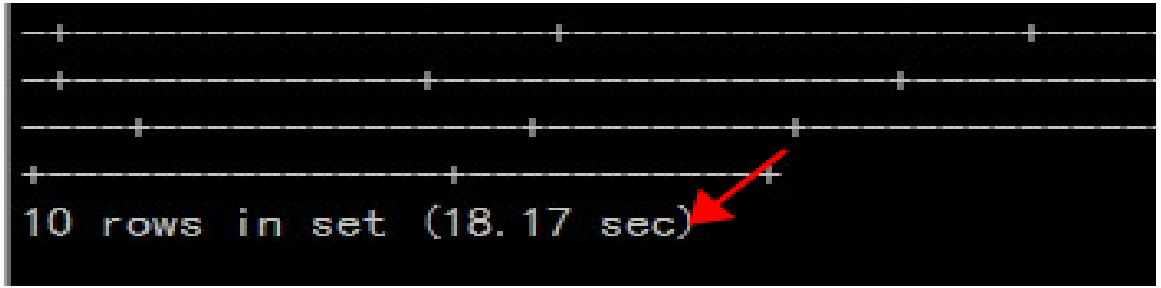
join_buffer_size = 1M

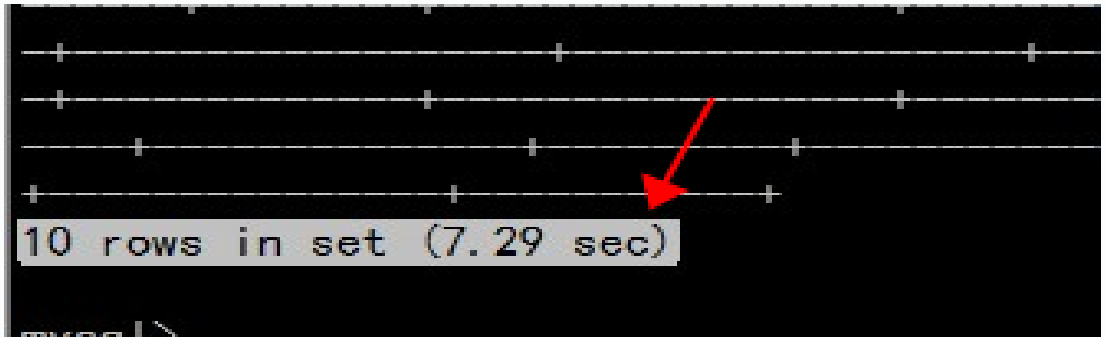
read_buffer_size = 1M

tmp_table_size =256K

测试验证

tmp_table_size由256K调整至128MB





排查思路：

- 查看SQL执行计划；
- 查看数据库版本和优化器规则；
- 对比参数，硬件设置；
- 查看网络延迟；

最佳实践：

- 保持数据库参数配置一致

目录 content

扫码观看大会视频

背景介绍

1、 某手机客户端上云

2、 第一次系统切割失败，数据库CPU 100%

3、 需要在第二次割接前排除原因

问题排除--跨版本升级(MySQL:5.5->5.6)

用户本地的mysql版本是5.5，而RDS的版本是5.6，用户的一条sql在本地 5.5执行只需要零点几秒，而在RDS上需要10多秒。

1) 5.5的优化器策略：

- index_merge=on,index_merge_union=on,index_merge_sort_union=on,index_merge_intersection=on,engine_condition_pushdown=on

2) 5.6的优化器策略：

- index_merge=on,index_merge_union=on,index_merge_sort_union=on,index_merge_intersection=on,engine_condition_pushdown=on,**block_nested_loop=on**.....

optimizer_switch： block_nested_loop=on

```
mysql> explain SELECT *
   ->  FROM t1 this_
   ->  LEFT OUTER JOIN t2 item2_ ON this_.itemId = gameitem2_.id
   ->  LEFT OUTER JOIN t3 group3_ ON gameitem2_.groupId =gamegroup3_.id
         …….
   ->  LEFT OUTER JOIN t8 leagueitem10_ ON leagueinfo7_.itemId =leagueitem10_.id
   ->  ORDER BY this_.id ASC LIMIT 20;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	this_	ALL	NULL	NULL	NULL	NULL	257312	Using temporary; Using filesort
1	SIMPLE	item2_	eq_ref	PRIMARY	PRIMARY	4	this_.itemId	1	NULL
1	SIMPLE	group3_	ALL	PRIMARY	NULL	NULL	NULL	6	Using where; **Using join buffer (Block Nested Loop)**

optimizer_switch： block_nested_loop=off

mysql> set optimizer_switch='....block_nested_loop=off....';

```
+----+-------------+---------------+--------+---------------+---------+---------+---------------------------+------+-------+
| id | select_type | table         | type   | possible_keys | key     | key_len | ref                       | rows | Extra |
+----+-------------+---------------+--------+---------------+---------+---------+---------------------------+------+-------+
|  1 | SIMPLE      | this_         | index  | NULL          | PRIMARY | 4       | NULL                      |   20 | NULL  |
|  1 | SIMPLE      | item2_        | eq_ref | PRIMARY       | PRIMARY | 4       | this_.itemId              |    1 | NULL  |
|  1 | SIMPLE      | group3_       | eq_ref | PRIMARY       | PRIMARY | 4       | item2_.groupId            |    1 | NULL  |
```

问题排除 — 字符串存储时间导致隐士转换

```
CREATE TABLE `test_date` (
    `id` int(11) DEFAULT NULL,
    `gmt_create` varchar(100) DEFAULT NULL,
    KEY `ind_gmt_create` (`gmt_create`)
) ENGINE=InnoDB AUTO_INCREMENT=524272;
```

5.5版本

```
mysql> explain  select * from test_date where gmt_create BETWEEN DATE_ADD(NOW(), INTERVAL - 1 MINUTE) AND
DATE_ADD(NOW(), INTERVAL 15 MINUTE) ;
+----+-------------+-----------+-------+---------------+---------------+---------+------+------+-------------+
| id | select_type | table     | type  | possible_keys | key           | key_len | ref  | rows | Extra       |
+----+-------------+-----------+-------+---------------+---------------+---------+------+------+-------------+
|  1 | SIMPLE      | test_date | range | ind_gmt_create | ind_gmt_create | 303     | NULL |    1 | Using where |
```

扫码观看大会视频

问题排除 — 字符串存储时间导致隐士转换

```
CREATE TABLE `test_date` (
    `id` int(11) DEFAULT NULL,
    `gmt_create` varchar(100) DEFAULT NULL,
    KEY `ind_gmt_create` (`gmt_create`)
) ENGINE=InnoDB AUTO_INCREMENT=524272;
```

5.6版本

```
mysql> explain select * from test_date where gmt_create BETWEEN DATE_ADD(NOW(), INTERVAL - 1 MINUTE) AND
DATE_ADD(NOW(), INTERVAL 15 MINUTE) ;
+----+-------------+-----------+------+---------------+------+---------+------+---------+-------------+
| id | select_type | table     | type | possible_keys | key  | key_len | ref  | rows    | Extra       |
+----+-------------+-----------+------+---------------+------+---------+------+---------+-------------+
|  1 | SIMPLE      | test_date | ALL  | ind_gmt_create | NULL | NULL    | NULL | 2849555 | Using where |
+----+-------------+-----------+------+---------------+------+---------+------+---------+-------------+
| Warning | 1739 | Cannot use range access on index 'ind_gmt_create' due to type  on field 'gmt_create'
```

扫码观看大会视频

排查思路：

- 分析SQL执行计划；
- 对比数据库版本和优化器规则

最佳实践：

- 保持数据库版本一致
- 功能和性能测试缺一不可

目录 content

扫码观看大会视频

背景介绍

- 1、 某APP应用上云后数据库CPU 100%，系统回滚会出现数据丢失
- 2、 弹性升级需要时间较长，要在白天业务高峰来临之际消除故障

问题排除—规格配置较小

用户本地物理机的配置是云上RDS的规格两倍，导致慢SQL出现堆积

1 本地物理机配置：

2U机箱，2*Intel E5-2609 v2 4核， 内存：64G ；磁盘ssd，Raid5；

2 RDS的配置:

逻辑cpu8核，内存32G，最大IOPS: 12000

紧急救援-优化SQL

```
mysql> explain  SELECT id FROM test WHERE status=30 and delStatus = 0 and publicStatus = 2 and
audit = 1  ORDER BY finishTime desc LIMIT 20
           id:  1
  select_type: SIMPLE
        table:  test
         type:  index_merge
possible_keys: Index_public,idx_delStatus
          key:  Index_public,idx_delStatus
      key_len:  4,4
         rows:  30137696
```

Extra: Using intersect(Index_public,idx_delStatus); Using where; Using filesort

索引情况：

```
PRIMARY KEY (`id`),
KEY `Index_public` (`publicStatus`),
KEY `index_finishTime` (`finishTime`),
KEY `idx_delStatus` (`delStatus`),
```

扫码观看大会视频

紧急救援-优化SQL

1.SQL的执行计划性能较低，走了两个索引的**intersect**，需要计算大量的数据rows: 30137696。

2.第一种解决办法是控制优化器的策略；第二种办法让表走**index_finishTime` (`finishTime`)** .

3.采取第二种办法将**idx_delStatus**索引删除，索引删除后执行计划恢复正常，性能急速提升：

紧急救援-优化SQL


```
mysql>explain  SELECT id FROM test WHERE status=30 and delStatus = 0 and publicStatus = 2 and audit = 1  ORDER BY finishTime desc LIMIT 20
```

```
           id:  1
  select_type: SIMPLE
        table:  test
         type:  index
possible_keys: Index_public
          key:  index_finishTime
      key_len:  8
         rows:  40
        Extra: Using where;
```

排查思路：

- 对比数据库资源配置；
- 分析SQL执行计划

最佳实践：

- 保持数据库资源配置一致
- 功能和性能测试缺一不可

目录 content

扫码观看大会视频

背景介绍

- 1、 某大型系统从Oracle迁移到RDS MySQL
- 2、 迁移到RDS后出现 CPU 100%，需要紧急解决

原因分析-子查询

```
SELECT first_name   FROM employees
    WHERE emp_no IN
    (SELECT emp_no FROM salaries_2000 WHERE salary = 5000);
```

MySQL的处理逻辑是遍历employees表中的每一条记录，代入到子查询中中去

改写子查询

```
SELECT first_name  FROM employees emp,
    (SELECT emp_no FROM salaries_2000 WHERE salary = 5000) sal
    WHERE emp.emp_no = sal.emp_no;
```

执行时间：1200S→0.1S

最佳实践：

- 子查询在5.1，5.5版本中都存在较大风险，将子查询改为关联
- 使用Mysql 5.6的版本，可以避免子查询的问题

目录 content

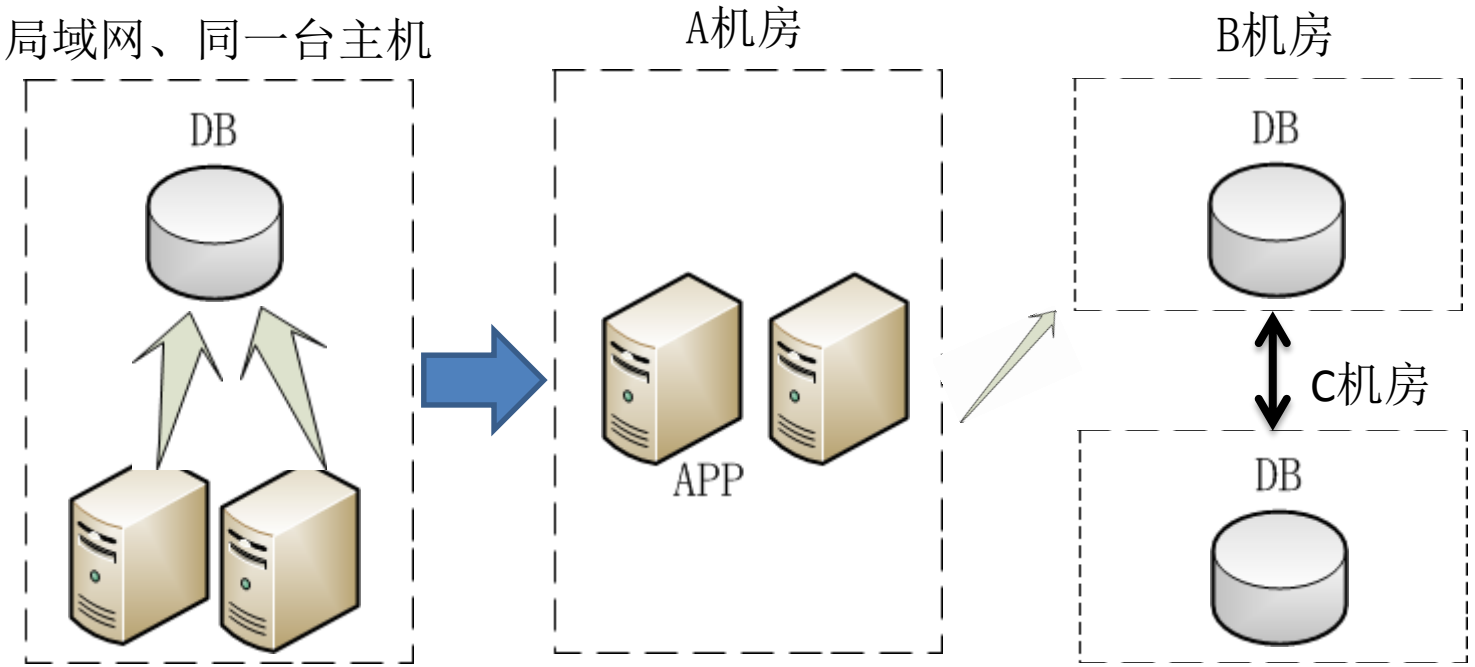
扫码观看大会视频

背景介绍

1、 某电商系统迁移上云测试过程中发现性能较低

2、 应用代码，数据库配置完全一样

原因分析-网络延迟放大



最佳实践：

- 需要考虑上云后网络延迟对性能的影响，优化应用与数据库的访问
- 应用和数据库尽量保持在同一个可用区内访问

总结

配置保持一致：版本，参数，规格等

考虑网络延迟：带宽，跨机房等

2016 The Computing Conference

THANKS