性能优化之永恒之道（1）（实时sql优化vs业务字段冗余vs离线计算）

原 荐

em_aaron 发布于 07/05 22:23 字数 2876 阅读 1964 收藏 55 点赞 6 评论 2

Navicat Lite Hadoop

在项目中，随着时间的推移，数据量越来越大，程序的某些功能性能也可能会随之下降，那么此时我们不得不需要对之前的功能进行性能优化。如果优化方案不得当，或者说不优雅，那可能将

对整个系统产生不可逆的严重影响。

此篇博主为大家分享一些根据自己多年的大数据分布式工作经验总结出优化的方案。

1.实时sql优化：就是将分析出来耗时的sql进行重写、拆分成多次查询后数据重组、去掉sql函数等等；sql能干的事情，程序肯定能干，且程序运行的性能一般情况会快很多，而且web服务器可以部署很多台；优点：可实现快速优化，且性能非常可观；缺点：可能会增加程序的复杂度；

2.业务冗余字段：就是在更新某张数据库业务表时，将其关联表的某些字段冗余进来，以减少某些业务的表关联查询从而达到提升速度效果；优点：实现简单；缺点：此方案只适用于某些比较简单的场景，如果关联的表非常多，且业务需要查询的字段也非常多，此时程序将对其它业务表的业务产生非常严重的侵入【即在更新别的表时候，需要更新该表】，在后期优化过程中，该方案还涉及到初始化数据的问题，另外数据一致性问题堪忧【如多业务保存中途宕机】，所以复杂业务不建议使用该方案。

3.离线计算：就是利用定时任务或者hadoop等到一些离线计算的技术，将数据跑成报表的方式。此方案适用于可支持非实时数据的查看的业务【如报表等等】。优点，可单独部署，不影响主程序，且性能优化效果非常可靠；缺点：数据非实时，且可能出现报表数据和真是数据某些字段不一致的情况，程序复杂度倍增且需要额外服务器支撑。

基于上诉的分析，博主最推荐的还是方案1和方案3这两种处理方案。在优化方案选择的优先级别上：

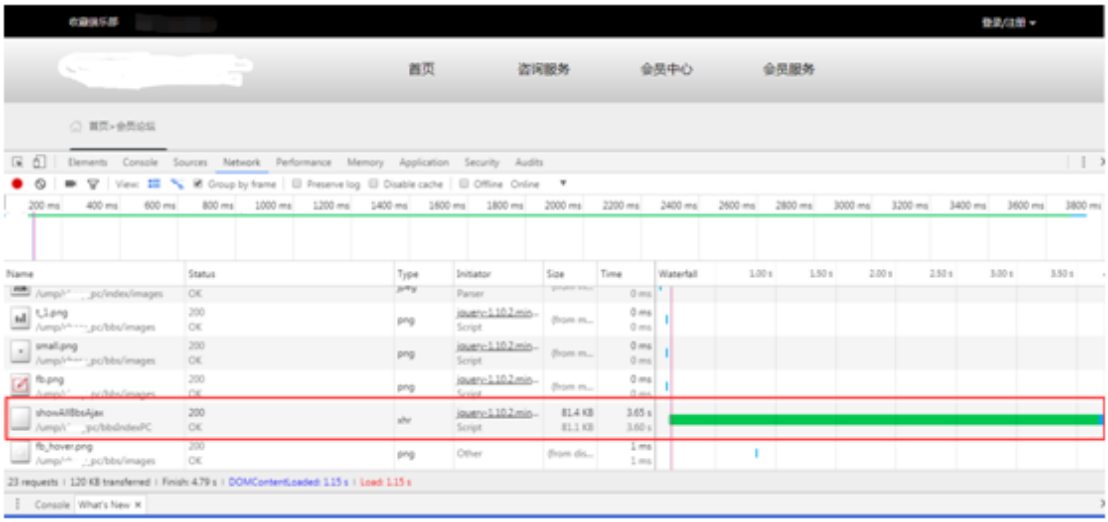
实时sql优化>离线计算>数据冗余；

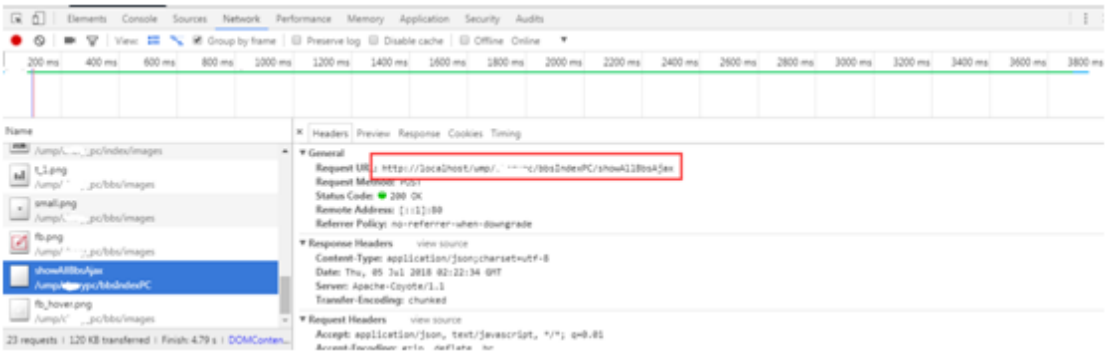
注意：在某些特定场景下【如业务非常简单时】，离线计算的优先级别小于数据冗余。

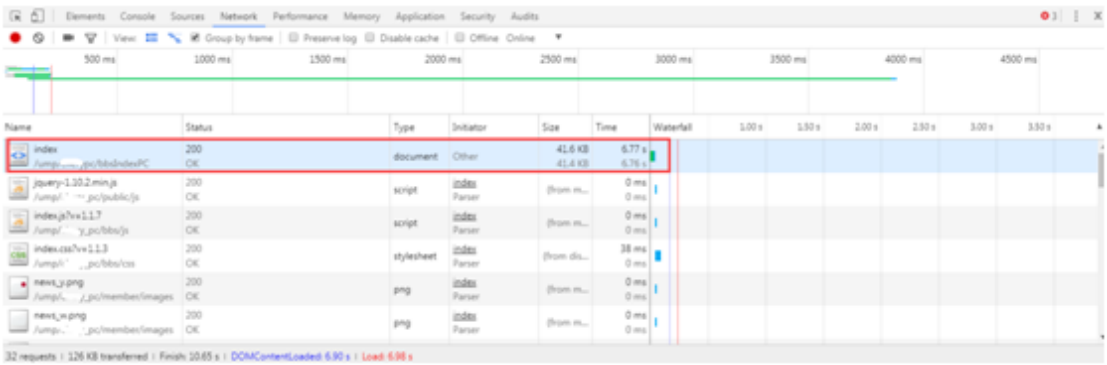
下面举例博主在实际项目中的一些sql优化例子

定位步骤[本次以后台sql导致的性能为例，其它如前端或程序处理导致性能问题不考虑]:

1.使用google浏览器按F12，打开性能差页面，检查耗时的请求







2.根据请求找到访问的controller–service–dbs–->sql

3.sql性能分析【将sql用生产数据库进行运行分析，此次就暂时不讲述使用explain分析的方法，下一篇再详细讲解】

【例1】.根据专区id或者板块id查询该专区或板块的发帖量，在navicat中运行结果27s

```
SELECT
        count(wcc_bbs_reply.id) counts
FROM
        wcc_bbs_reply
WHERE
        EXISTS (
                SELECT
                        *
                FROM
                        (
```

```
                                SELECT
                                        wcc_bbs.id bbs_id
                                FROM
                                        wcc_bbs
                                WHERE
                                        (
                                                wcc_bbs.refer_id = 2
                                                AND wcc_bbs.type = 4
                                                AND wcc_bbs.is_delete = 0
                                                AND wcc_bbs.bbs_state = '2'
                                        )
                                UNION ALL
                                        SELECT
                                                wcc_bbs.id bbs_id
                                        FROM
                                                wcc_bbs,
                                                wcc_bbs_area
                                        WHERE
                                                (
                                                        wcc_bbs.type = 1
                                                        AND wcc_bbs.refer_id =
wcc_bbs_area.id
                                                        AND wcc_bbs_area.bbs_section
= 2
                                                        AND wcc_bbs.is_delete = 0
                                                        AND wcc_bbs.bbs_state = '2'
                                                )
                        ) bbs_extend
                WHERE
                        bbs_extend.bbs_id = wcc_bbs_reply.bbs
        )
```

经过语义分析，该条sql其实是想查询id为2的板块与它下面的专区对应的回帖量总和。那我们是否就可以将该条sql拆分为直接挂在id为2的板块下的帖子回复量和挂在id为2的板块下的专区上的帖子回复量呢？答案是当然可以的，经拆分：

第一条sql（执行时间0.022s）：

```
SELECT
        count(wcc_bbs_reply.id) counts
FROM
        wcc_bbs_reply ,  wcc_bbs
WHERE
        wcc_bbs.refer_id =2
        AND wcc_bbs.type = 4
        AND wcc_bbs.is_delete = 0
        AND wcc_bbs.bbs_state = '2'
  and wcc_bbs.id = wcc_bbs_reply.bbs
```

第二条sql（执行时间0.17s）：

```
SELECT
        count(wcc_bbs_reply.id) counts
```

```
FROM
        wcc_bbs_reply,
        (
                SELECT
                        wcc_bbs.id bbs_id
                FROM
                        wcc_bbs,
                        wcc_bbs_area
                WHERE
                        (
                                wcc_bbs.type = 1
                                AND wcc_bbs.refer_id = wcc_bbs_area.id
                                AND wcc_bbs_area.bbs_section = 2
                                AND wcc_bbs.is_delete = 0
                                AND wcc_bbs.bbs_state = '2'
                        )
        ) a
WHERE
        a.bbs_id = wcc_bbs_reply.bbs
```

由此可分析出上诉经过上诉拆分后性能可提高数百倍，小伙伴们是不是特别兴奋；

【例2】.后台帖子管理列表分页查询功能，执行时间为29s，**优化方案-性能逐节衰减方案**

```
SELECT
        *
FROM
        (
                SELECT
                        a.id,
                        '' AS chId,
                        a.bbs_title,
                        ifnull(c.member_name, 'admin') AS member_name,
                        d.id AS section,
                        d. NAME AS sectionName,
                        b.id AS area,
                        b. NAME AS areaName,
                        a.bbs_publish_time,
                        a.bbs_last_reply_time,
                        a.bbs_reply_num,
                        a.bbs_read_num,
                        a.bbs_agree_num,
                        a.bbs_state,
                        a.bbs_lock,
                        a.bbs_elite,
                        a.bbs_label_text,
                        a.bbs_top,
                        a.is_delete AS isdelete,
                        a.bbs_property AS proerty,
                        a.type AS type,
                        a.bbs_close AS CLOSE,
                        a.plate_bbs_top AS plateTop,
                        a.pcidx_bbs_show AS pcidxBbsShow
                FROM
                        wcc_bbs a
                LEFT JOIN wcc_bbs_area b ON a.refer_id = b.id
```

```
LEFT JOIN wcc_ch_member c ON a.ch_member = c.id
LEFT JOIN wcc_bbs_section d ON b.bbs_section = d.id
WHERE
        a.type = '1'
UNION ALL
        SELECT
                a.id,
                '' AS chId,
                a.bbs_title,
                ifnull(c.member_name, 'admin') AS member_name,
                d.id AS section,
                d. NAME AS sectionName,
                '' AS area,
                '' AS areaName,
                a.bbs_publish_time,
                a.bbs_last_reply_time,
                a.bbs_reply_num,
                a.bbs_read_num,
                a.bbs_agree_num,
                a.bbs_state,
                a.bbs_lock,
                a.bbs_elite,
                a.bbs_label_text,
                a.bbs_top,
                a.is_delete AS isdelete,
                a.bbs_property AS proerty,
                a.type AS type,
                a.bbs_close AS CLOSE,
                a.plate_bbs_top AS plateTop,
                a.pcidx_bbs_show AS pcidxBbsShow
        FROM
                wcc_bbs a
        LEFT JOIN wcc_ch_member c ON a.ch_member = c.id
        LEFT JOIN wcc_bbs_section d ON a.refer_id = d.id
        WHERE
                a.type = '4'
        UNION ALL
                SELECT
                        a.id,
                        b.id AS chId,
                        a.bbs_title,
                        ifnull(c.member_name, 'admin') AS
member_name,
                        b.id AS section,
                        b. NAME AS sectionName,
                        '' AS area,
                        '' AS areaName,
                        a.bbs_publish_time,
                        a.bbs_last_reply_time,
                        a.bbs_reply_num,
                        a.bbs_read_num,
                        a.bbs_agree_num,
                        a.bbs_state,
                        a.bbs_lock,
                        a.bbs_elite,
                        a.bbs_label_text,
                        a.bbs_top,
                        a.is_delete AS isdelete,
```

```
                a.bbs_property AS proerty,
                a.type AS type,
                a.bbs_close AS CLOSE,
                a.plate_bbs_top AS plateTop,
                a.pcidx_bbs_show AS pcidxBbsShow
        FROM
                wcc_bbs a
        LEFT JOIN wcc_bbs_circle b ON a.refer_id = b.id
        LEFT JOIN wcc_ch_member c ON a.ch_member = c.id
        WHERE
                a.type <> '1'
        AND a.type <> '4'
) d
WHERE
        1 = 1
AND d.isdelete = 0
ORDER BY
        d.bbs_publish_time DESC
LIMIT 10
```

语义分析：上面这条sql中包含了两个union all,也句是由3条sql结果合并组成；既然功能完全是独立的，我们是否就可以分三次进行查询组合呢，这么复杂的sql，简单化多好，看着也舒服有么有；但是大家别被博主玩坏了，这里可是有排序的，且为分页查询，问题可不是分sql查询这么简单；难道就没办法解决了吗？这就是今天分享的一大SQL经典优化案例，sql性能衰减方案，该方案就是先根据当前页*每页大小（该值名称定义为total，以下将以此名称表示），然后将多个子句按照上诉得出来的total值查询出total条数据，然后进行排序取出当前查询的那页的数据。即：

子句1.【执行时间0.22s】

```
SELECT
        a.id,
        '' AS chId,
        a.bbs_title,
        ifnull(c.member_name, 'admin') AS member_name,
        d.id AS section,
        d. NAME AS sectionName,
        b.id AS area,
        b. NAME AS areaName,
        a.bbs_publish_time,
        a.bbs_last_reply_time,
        a.bbs_reply_num,
        a.bbs_read_num,
        a.bbs_agree_num,
        a.bbs_state,
        a.bbs_lock,
        a.bbs_elite,
        a.bbs_label_text,
        a.bbs_top,
        a.is_delete AS isdelete,
        a.bbs_property AS proerty,
        a.type AS type,
        a.bbs_close AS CLOSE,
```

```
        a.plate_bbs_top AS plateTop,
        a.pcidx_bbs_show AS pcidxBbsShow
FROM
        wcc_bbs a
LEFT JOIN wcc_bbs_area b ON a.refer_id = b.id
LEFT JOIN wcc_ch_member c ON a.ch_member = c.id
LEFT JOIN wcc_bbs_section d ON b.bbs_section = d.id
WHERE
        a.type = '1'
AND a.is_delete = 0
ORDER BY
        a.bbs_publish_time DESC
LIMIT 10
```

子句2【执行时间0.20s】

```
SELECT
        a.id,
        '' AS chId,
        a.bbs_title,
        ifnull(c.member_name, 'admin') AS member_name,
        d.id AS section,
        d. NAME AS sectionName,
        '' AS area,
        '' AS areaName,
        a.bbs_publish_time,
        a.bbs_last_reply_time,
        a.bbs_reply_num,
        a.bbs_read_num,
        a.bbs_agree_num,
        a.bbs_state,
        a.bbs_lock,
        a.bbs_elite,
        a.bbs_label_text,
        a.bbs_top,
        a.is_delete AS isdelete,
        a.bbs_property AS proerty,
        a.type AS type,
        a.bbs_close AS CLOSE,
        a.plate_bbs_top AS plateTop,
        a.pcidx_bbs_show AS pcidxBbsShow
FROM
        wcc_bbs a
LEFT JOIN wcc_ch_member c ON a.ch_member = c.id
LEFT JOIN wcc_bbs_section d ON a.refer_id = d.id
WHERE
        a.type = '4'
AND a.is_delete = 0
ORDER BY
        a.bbs_publish_time DESC
LIMIT 10
```

子句3【执行时间0.21s】

```
SELECT
        a.id,
        b.id AS chId,
        a.bbs_title,
        ifnull(c.member_name, 'admin') AS member_name,
        b.id AS section,
        b. NAME AS sectionName,
        '' AS area,
        '' AS areaName,
        a.bbs_publish_time,
        a.bbs_last_reply_time,
        a.bbs_reply_num,
        a.bbs_read_num,
        a.bbs_agree_num,
        a.bbs_state,
        a.bbs_lock,
        a.bbs_elite,
        a.bbs_label_text,
        a.bbs_top,
        a.is_delete AS isdelete,
        a.bbs_property AS proerty,
        a.type AS type,
        a.bbs_close AS CLOSE,
        a.plate_bbs_top AS plateTop,
        a.pcidx_bbs_show AS pcidxBbsShow
FROM
        wcc_bbs a
LEFT JOIN wcc_bbs_circle b ON a.refer_id = b.id
LEFT JOIN wcc_ch_member c ON a.ch_member = c.id
WHERE
        a.type <> '1'
AND a.type <> '4'
AND a.is_delete = 0
ORDER BY
        a.bbs_publish_time DESC
LIMIT 10
```

由此可以看出上面三条sql查询出来耗时还不到一秒，组装是程序速度是非常快的，几乎时间可以忽略。但随着页数的增大，limit 后面的10就成当前页的大小的倍数增大，所以当前页越大时，需要查询运算的数据就会越来越多，性能就会进行衰减，且此方法会将数据load到内存占用内存空间，使用者需权衡使用。一个很好的解决到了一定页面后，性能衰减到客户不能承受的时间差时，我们可以判断页数到达多少页时，使用原来一条sql的查询，保证功能可使用。此方案适用于列表在业务上查询使用新数据多，老数据几乎不用的场景。

【例3】.会员明细列表查询,会统计很多用户信息，如积分等等；**经典left join查询优化方案**

```
SELECT
        m.id,
        m.member_name,
        m.phone_no,
        CASE
```

```
    WHEN m.sex = '1' THEN
            '男'
    WHEN m.sex = '2' THEN
            '女'
    END sex,
     m.email,
     CASE
    WHEN m.member_level = '0' THEN
            '游客'
    WHEN m.member_level = '1' THEN
            '普卡'
    WHEN m.member_level = '2' THEN
            '银卡'
    WHEN m.member_level = '3' THEN
            '金卡'
    END member_level,
     m.registration_time,
     m.expiration_time,
     s. NAME,
     d.dealer_name,
     d.dealer_no,
     n.vin_code,
     n.carType,
     j.canUseScore,
     j1.usedScore,
     j2.sumScore
    FROM
            wcc_ch_member m
    LEFT JOIN (
            SELECT
                    v.wcc_member_info,
                    GROUP_CONCAT(
                            t.vehicle_type_name SEPARATOR ';'
                    ) AS carType,
                    GROUP_CONCAT("'", v.vin_code, "'") AS vin_code
            FROM
                    wcc_ch_vehicle v,
                    wcc_vehicle_type t
            WHERE
                    v.wcc_vehicle_type = t.id
            AND (v. STATUS = 1 OR v. STATUS = 3)
            GROUP BY
                    v.wcc_member_info
    ) n ON n.wcc_member_info = m.id
    LEFT JOIN wcc_friend h ON h.open_id = m.open_id
    LEFT JOIN wcc_sale_assist s ON h.said = s.id
    LEFT JOIN wcc_ch_dealer d ON s.dealer = d.id
    LEFT JOIN (
            SELECT
                    wcc_member_info,
                    sum(can_use_score) canUseScore
            FROM
                    wcc_ch_integral_detail
            WHERE
                    isadd = 1
            AND can_use_score > 0
            AND (
                    expiration_Date IS NULL
```

```
                OR expiration_date >= '2018-07-05 14:00:34'
        )
        GROUP BY
                wcc_member_info
) j ON m.id = j.wcc_member_info
LEFT JOIN (
        SELECT
                wcc_member_info,
                sum(score - return_score) usedScore
        FROM
                wcc_ch_integral_detail
        WHERE
                isadd = '0'
        GROUP BY
                wcc_member_info
) j1 ON m.id = j1.wcc_member_info
LEFT JOIN (
        SELECT
                wcc_member_info,
                sum(score) sumScore
        FROM
                wcc_ch_integral_detail
        WHERE
                isadd = '1'
        GROUP BY
                wcc_member_info
) j2 ON m.id = j2.wcc_member_info
WHERE
        1 = 1
LIMIT 0,
 10
```

分析，此sql中全是left子句，线上功能已经处于不可用状态.查询表单如下：

页面查询表单分析，子句中只有一个vin码是查询条件，vin对应唯一的一辆车，一辆车又对应唯一的一个车主会员，于是我们可以将所有子句拆分出来分批查询组装后返回；思路如下，在没有vin查询时候，我们根据上面条件单表查询wcc_ch_member得出当前页的10条数据的id集合，然后根据这10条数据使用memberid in（.....）的方式分批次去查询后面的统计字段和其他子句的字段进行组装。经分析，所有子句查询都在0.01秒左右，且wcc_ch_member在单表查询是也是0.02秒左右，由此可推算出第一种情况查询优化效果提升千万倍性能。那么在有vin码条件的情况下呢，我们首先可以根据vin码查询出member表的会员id,然后根据id再去按照上面的方法查询对应的统计，分析出来只多了一步member查询，性能上和上诉无vin码查询是效果几乎无变化，且可能更快。

博主今天的分享就到这里。最后总结一下，所有子句的查询均可以分批次查询来优化，所有left join类型的查询其实均可转为单表查询。当然，sql业务拆分级的优化方案不止上诉几种，这里博主只是抛砖头引璞玉。如果你对业务级拆分性能优化方案或者其它服务器性能优化方案感兴趣的话，请点赞博主，并欢迎同博主交流。

em_aaron
粉丝 17 博文 24 码字总数 32281 作品 3
黄浦 高级程序员
关注

评论(2)

em_aaron 今天 11:43
不会的，数据库都是可以集群的，而且你一次请求才0.01秒，
回复 举报

青青子衿1111 今天 10:31
这样拆分出来 会不会增加对数据库连接所产生的开销负担?
回复 举报

在这里发表你对此文的观点

插入表情 # 插入软件 0/250 发表评论

相关文章 最新文章

Mysql性能优化一

mysql的性能优化无法一蹴而就，必须一步一步慢慢来，从各个方面进行优化，最终性能就会有大的提升。Mysql数据库的优化技术 对mysql优化是一个综合性的技术，主要包括 表的设计合理化(符合3...

JAVA_NINA · 2016/05/09 · 0

spark2.0新特性

使用： 离线计算：数据源大多来自hdfs(hive)，所以sql使用的非常多，几乎每个离线计算作业都会用到hivecontext或sqlcontext 实时计算：streaming模块 图计算在企业里用的很少，需求少 数据挖...

曾晓森 · 2016/09/18 · 0

hibernate与数据库建模--原作robbin

其实围绕Hibernate的话题，我都已经说过不下30遍，以致于最近两年以来，我对所有Hibernate的问题都不愿意再回应。另外最近一年多来，使用Rails的ActiveRecord，让我对ORM的认识又加深了很多，...

ChowJames · 2012/09/02 · 0

如何打造千万级Feed流系统？阿里数据库技术解读

2017年的双十一又一次刷新了记录，交易创建峰值32.5万笔/秒、支付峰值25.6万笔/秒。而这样的交易和支付等记录，都会形成实时订单Feed数据流，汇入数据运营平台的主动服务系统中去。数据运...

天池大数据科研平台 · 2017/12/04 · 0

数据库案例集锦 – 开发者的《如来神掌》

标签 PostgreSQL , PG DBA cookbook , PG Oracle兼容性 , PG 架构师 cookbook , PG 开发者 cookbook , PG 应用案例 背景 「剑魔独孤求败，纵横江湖三十馀载，杀尽仇寇，败尽英雄，天下更无抗...

德哥 · 2017/06/09 · 0

Mysql 性能优化——必胜之道

mysql的性能优化是运维和DBA们常常面对的问题，也是各大公司招聘人才时看中的要点之一。性能优化听上去很难，似乎只有大神才能做，然而，mysql的性能优化绝不是运维独自一个能完成的，DBA、开...

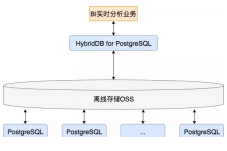
32氪 · 2017/07/31 · 0

Mysql 性能优化——必胜之道

mysql的性能优化是运维和DBA们常常面对的问题，也是各大公司招聘人才时看中的要点之一。性能优化听上去很难，似乎只有大神才能做，然而，mysql的性能优化绝不是运维独自一个能完成的，DBA、开...

super李导 · 2017/03/14 · 0

PgSQL · 最佳实践 · 双十一数据运营平台订单Feed数据洪流实时分析方案



摘要 2017年的双十一又一次刷新了记录，交易创建峰值32.5万笔/秒、支付峰值25.6万笔/秒。而这样的交易和支付等记录，都会形成实时订单Feed数据流，汇入数据运营平台的主动服务系统中去。 数据...

阿里云RDS–数据库内核组 · 2017/11/08 · 0

大数据下的数据分析平台架构

时间：2011-08-15 14:59 作者：谢超 随着互联网、移动互联网和物联网的发展，谁也无法否认，我们已经切实地迎来了一个海量数据的时代，数据调查公司IDC预计2011年的数据总量将达到1.8万亿GB，...

长征2号 · 2017/04/07 · 0

【干货合集】技无止境——2017双11核心技术大揭秘

一、视频分享 1、双11万亿流量下的分布式缓存Tair技术揭秘 视频主要从Tair发展和应用开始谈起，接着谈及双11面临的挑战，重点分享了性能优化方面的实践，最后对缓存难题给出了解决方案。 点击...

阿里云云栖社区 · 01/03 · 0

加载更多