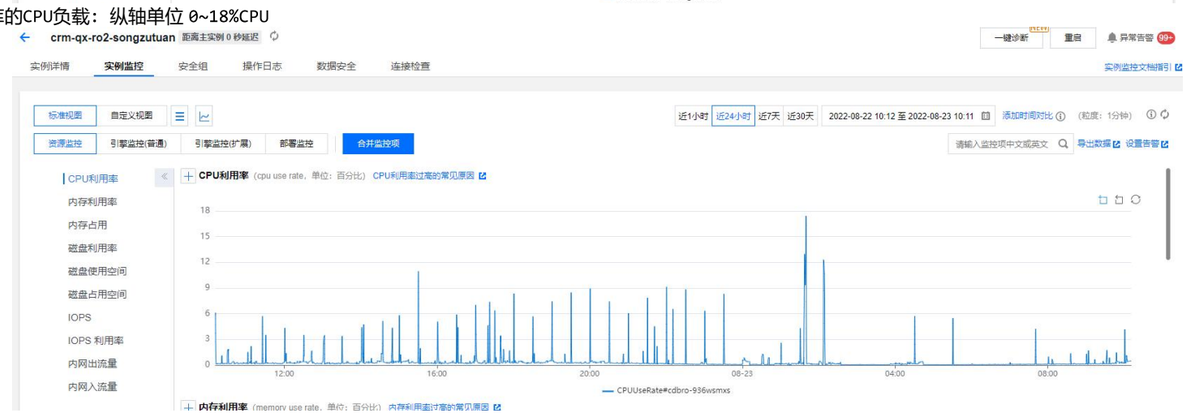
主从负载不均衡排查记录

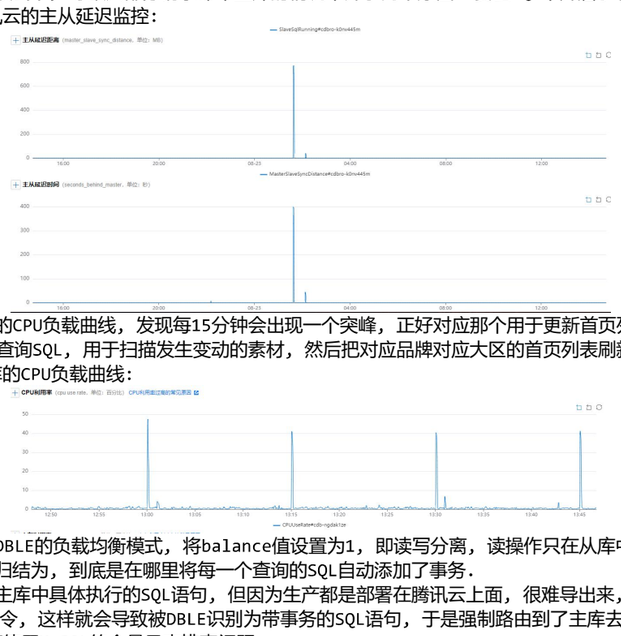
2022年8月24日 18:49

异常现象：一主二从的数据库集群，却经常出现主库满载而从库空闲的情况，虽然代码里面也存在一部分慢查询SQL，但也不可能都命中的主库.

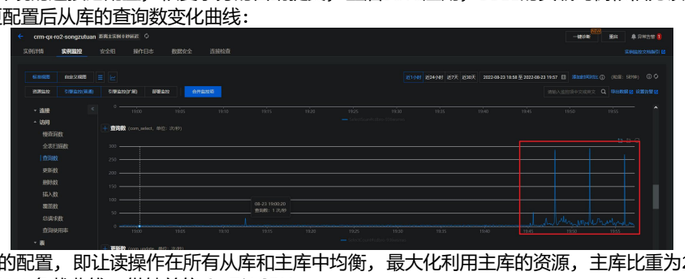
- 主库的CPU负载：纵轴单位 0~80%CPU
  - 

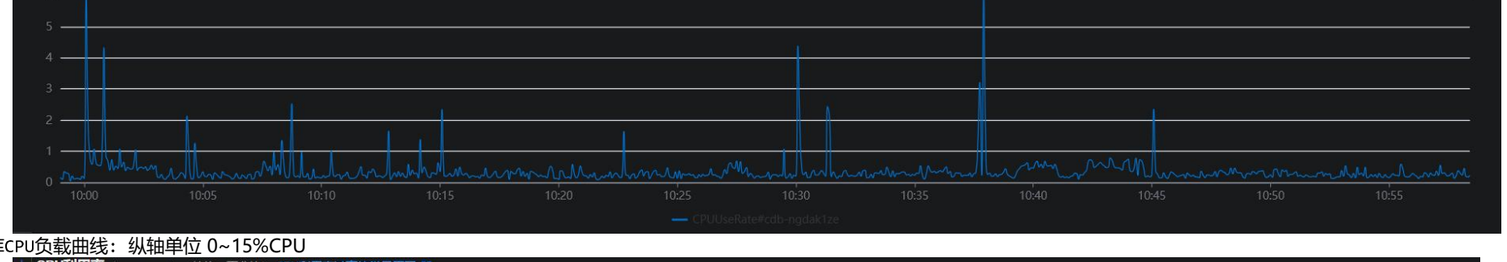
- 从库的CPU负载：纵轴单位 0~18%CPU
  - 

排查经过：

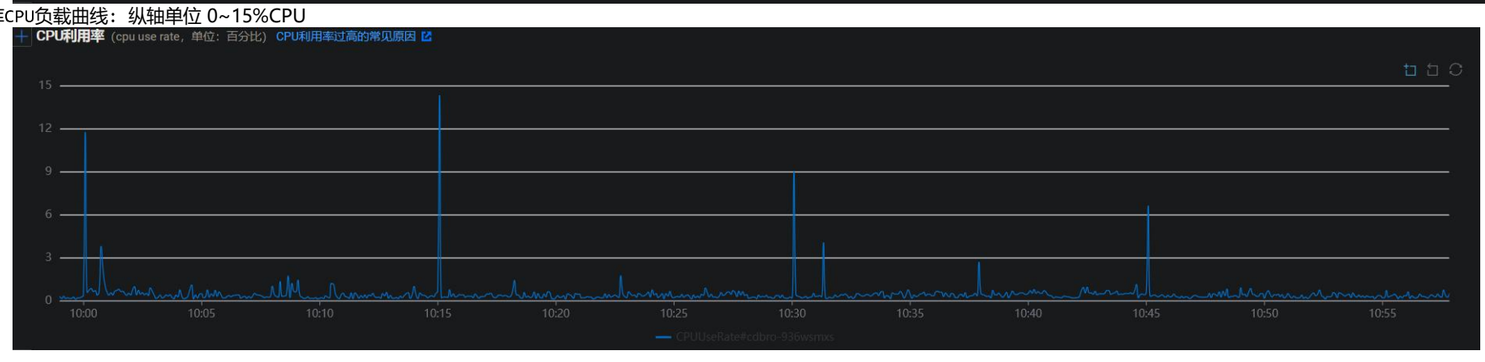
1. 生产环境的数据库集群是被中间件DBLE托管的，检查DBLE的配置，其中决定 负载均衡 模式的关键性参数：balance 的值为2，即在所有读库和写库中均衡，并且两个读库(从库)的比重设置为3，一个写库(主库)的比重设置为2，没有问题.
   - 配置文件schema.xml：
```
<dataHost balance="2" maxCon="1000" minCon="10" name="qx_mall_dn1_host" slaveThreshold="100">
    <heartbeat>show slave status</heartbeat>
    <writeHost host="hostM1" url="10.242.0.197:3306" password="" user="dble_user" disabled="false" id="hostM1" weight="2">
        <readHost host="hostS1" url="10.242.0.208:3306" password="" user="dble_user" disabled="false" id="hostS1" weight="3" />
        <readHost host="hostS2" url="10.242.0.209:3306" password="" user="dble_user" disabled="false" id="hostS2" weight="3" />
    </writeHost>
</dataHost>
```
2. 负载均衡模式下，导致大部分请求命中主库的情况不外乎两个原因：要么SQL中开启了事务，要么存在主库-从库同步延迟的情况，当延迟超过一定阈值时，请求会被强制打到主库．但是通过腾讯云平台的查询，白天并没有出现主从延迟的情况.
   - 腾讯云的主从延迟监控：
   - 

3. 观察主库的CPU负载曲线，发现每15分钟会出现一个突峰，正好对应那个用于更新首页列表缓存的定时任务，也是每15分钟一次．从库这边，并没有出现每15分钟的突峰，说明定时任务的SQL没有命中从库，一直走的主库．据此，专门重点检查了这个定时任务的代码，发现确实没有任何事务，全是查询SQL，用于扫描发生变动的素材，然后把对应品牌对应大区的首页列表刷新到Redis缓存中，没有任何更新或者插入数据库的操作，基本可以确定不是在代码中手动添加了事务的原因.
   - 主库的CPU负载曲线：
   - 

4. 尝试修改DBLE的负载均衡模式，将balance值设置为1，即读写分离，读操作只在从库中均衡，只有写操作和开启了事务的读操作才会分发到主库．观察一段时间后发现，主库的CPU负载还是没降下来，并且每15分钟的突峰还在，说明这个定时任务的查询SQL里面必定被添加了事务．现在的问题可以归结为，到底是在哪里将每一个查询的SQL自动添加了事务.
5. 尝试导出主库中具体执行的SQL语句，但因为生产都是部署在腾讯云上面，很难导出来，于是尝试先在测试的数据库开启全量日志，手动触发那个每15分钟一次的定时任务，导出此时的数据库全量日志进行分析．从日志中找到定时任务的查询SQL，发现确实每条SQL后面都追加了一条commit命令，这样就会导致被DBLE识别为带事务的SQL语句，于是强制路由到了主库去执行．由此可以排除中间件DBLE的影响，问题归结到我们的Java应用为何会在每条SQL语句后面都自动追加了一条commit命令.
   - 如何使用MySQL的全量日志排查问题：
     i. 查看相关配置项：show global variables like 'gen%';
     ii. 设置全量日志的导出文件路径：set global general_log_file = '/data/mysql/db/20220823-hx-01.log';
     iii. 开启全量日志：SET GLOBAL general_log = 1;
     iv. 触发相关代码的执行;
     v. 关闭全量日志：SET GLOBAL general_log = 0;
     vi. 登录服务器，下载日志文件到本地进行分析.
   - 全量日志截图：
```
2022-08-23T16:03:49.878956+08:00     5270 Query SELECT
        shr.id,
        shr.title,
        shr.share_doc,
        shr.`status`,
        shr.brand_no,
        shr.org_code,
        shr.brand_code,
        shr.create_user,
        shr.update_user,
        shr.cover_url,
        shr.cover_forward_url,
        shr.start_time,
        shr.end_time,
        shr.create_time,
        shr.update_time,
        shr.material_type,
        shr.create_source,
        shr.create_type,random_share_init,random_view_init,
        shr.is_valid,
        shr.long_term_effective,
        shr.open_status,
        shr.link_first,
        shr.org_name,
        shr.forward_appid,
        shr.forward_appid_path,
        shr.gh_id,
        shr.create_job_number,
        shr.update_job_number

        FROM sgc_nobrand_brands snb
            LEFT JOIN sgc_hot_recommend shr ON snb.hot_recommend_id = shr.id
        WHERE snb.is_valid = 1
            AND shr.is_valid = 1
            AND snb.home_display = 1
            AND snb.brand_code = 'NOBRAND'

            AND (shr.org_code IN ('ALL','Z3')
                OR shr.org_code = 'I')


            and snb.blong_brand = 1

            AND shr.long_term_effective != 1
            AND '2022-08-23 00:30:00.637'  <  shr.start_time
            AND shr.start_time  <=  NOW()
            AND shr.open_status = 1

        limit 1
2022-08-23T16:03:49.885162+08:00     5270 Query commit
```
6. 怀疑是自动提交事务的问题，分别查了下 连接池，MyBatis，MySQL 有关自动提交的配置，发现MySQL和MyBatis都用的默认，即自动提交，而连接池居然关闭了自动提交，但代码里又没有手动去提交，这样很可能导致了，Java应用发起的每一条SQL都开启了事务，而又去提交事务，但是底层的MySQL又配置了自动提交，所以造成Java应用发起的每一条SQL，MySQL都自动为其追加了commit命令．为了证实这个猜想，把测试环境连接池的自动提交开启了，再次开启定时任务，导出全量日志，发现多余的commit消失了，证明确实是自动提交配置不统一的问题.
   - 连接池配置截图：
```
druid:
  initial-size: 10
  min-idle: 10
  max-active: 20
  max-wait: 20000
  time-between-eviction-runs-millis: 20000
  min-evictable-idle-time-millis: 300000
  max-evictable-idle-time-millis: 1800000
  validation-query: SELECT 'x'
  default-auto-commit: false
  test-while-idle: true
  test-on-borrow: false
  test-on-return: false
  keep-alive: true
  filter:
    wall:
      config:
        multi-statement-allow: true
  filters: stat,log4j2
```
7. 修改生产环境的连接池配置，恢复事务的自动提交，重启Java应用，DBLE的负载均衡依旧为读写分离，观察数据库的变化，发现主库的查询数立马降下去，从库的查询数也升上来了，解决了读写分离模式下，查询操作的SQL不走从库的问题.
   - 变更配置后从库的查询数变化曲线：
   - 
8. 恢复DBLE的配置，即让读操作在所有从库和主库中均衡，最大化利用主库的资源，主库比重为2，从库比重为3，观察一段时间发现主从库的负载基本跟比重一致，并且都能观察到每15分钟一次的突峰负载(即那个定时任务)，问题到此彻底解决.
   - 主库CPU负载曲线：纵轴单位 0~7%CPU
   -



- 从库CPU负载曲线：纵轴单位 0~15%CPU



排查结果：DBLE和MySQL的配置都没问题，业务代码也没问题，属于Java项目的基础配置有问题，不应该取消掉数据库连接池的自动提交设置.

相关建议：强烈建议在 测试环境 也部署一套数据库的主从集群, 并且被中间件DBLE托管, 与生产环境保持一致, 不然像这次直接在生产环境进行相关配置的摸索, 风险其实是很大的, 需要尽量避免.