

## 17 | 大厂都是怎么做MySQL to Redis同步的？

2020-04-04 李玥

后端存储实战课

[进入课程 >](#)



讲述：李玥

时长 14:17 大小 13.09M



你好，我是李玥。

之前我们在《[11 | MySQL 如何应对高并发（一）：使用缓存保护 MySQL](#)》这一节课中，讲到了 Read/Write Through 和 Cache Aside 这几种更新缓存的策略，这几种策略都存在缓存穿透的可能，如果缓存没有命中，那就穿透缓存去访问数据库获取数据。

一般情况下，只要我们做好缓存预热，这个缓存的命中率很高，能穿透缓存打到数据库上的请求比例就非常低，这些缓存的策略都是没问题的。但是如果说，我们的 Redis 缓存是一个超大规模的系统，那就又不一样了。



今天这节课，我们来说一下，在超大规模系统中缓存会面临什么样的问题，以及应该使用什么样的策略来更新缓存。

## 缓存穿透：超大规模系统的不能承受之痛

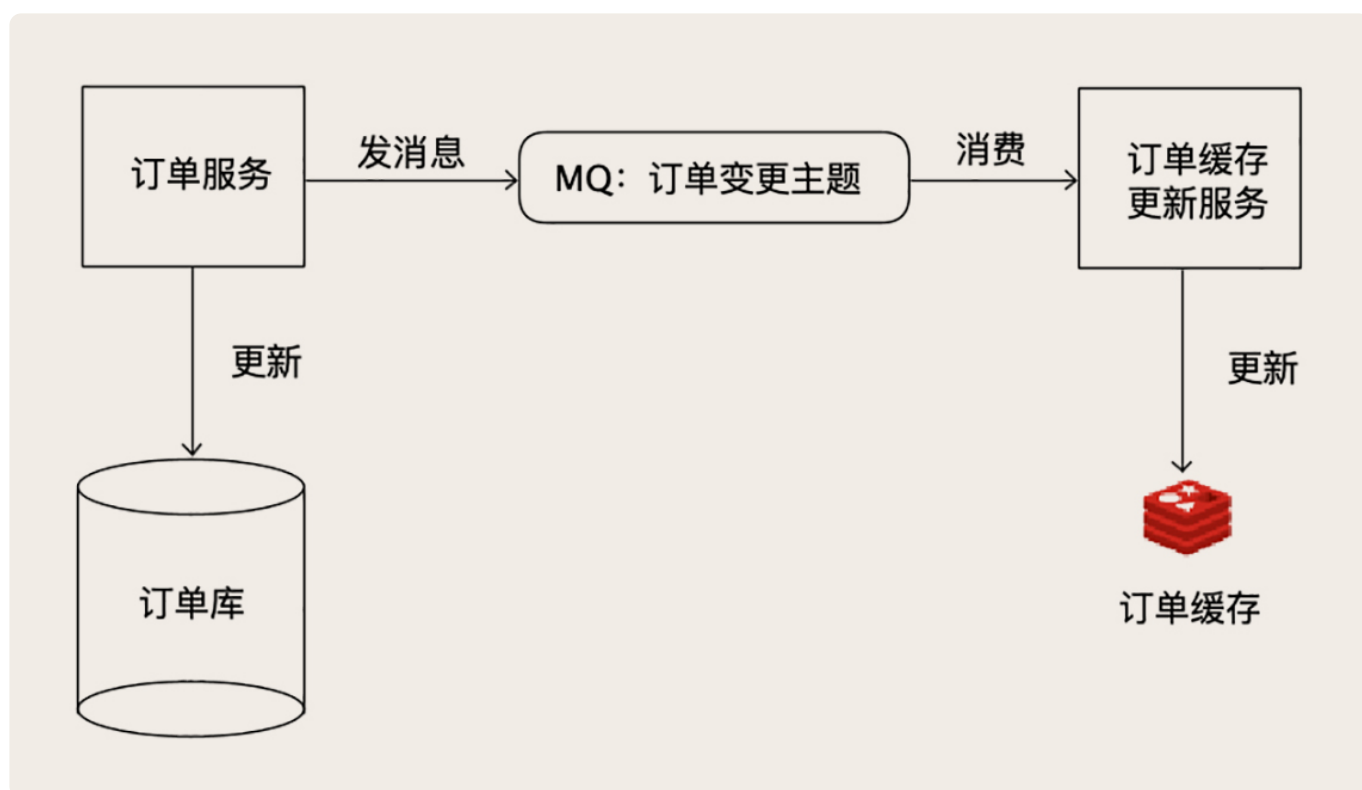
我们上节课讲到了如何构建 Redis 集群，由于集群可以水平扩容，那只要集群足够大，理论上支持海量并发也不是问题。但是，因为并发请求的数量这个基数太大了，即使有很小比率请求穿透缓存，打到数据库上请求的绝对数量仍然不小。加上大促期间的流量峰值，还是存在缓存穿透引发雪崩的风险。

那这个问题怎么解决呢？其实方法你也想得到，不让请求穿透缓存不就行了？反正现在存储也便宜，只要你买得起足够多的服务器，Redis 集群的容量就是无限的。不如把全量的数据都放在 Redis 集群里面，处理读请求的时候，干脆只读 Redis，不去读数据库。这样就完全没有“缓存穿透”的风险了，实际上很多大厂它就是这么干的。

在 Redis 中缓存全量的数据，又引发了一个新的问题，那就是，如何来更新缓存中的数据呢？因为我们取消了缓存穿透的机制，这种情况下，从缓存读到数据可以直接返回，如果没读到数据，那就只能返回错误了！所以，当系统更新数据库的数据之后，必须及时去更新缓存。

说到这儿，又绕回到那个老问题上了：怎么保证 Redis 中的数据和数据库中的数据同步更新？我们之前讲过用分布式事务来解决数据一致性的问题，但是这些方法都不太适合用来更新缓存，**因为分布式事务，对数据更新服务有很强的侵入性**。我们拿下单服务来说，如果为了更新缓存增加一个分布式事务，无论我们用哪种分布式事务，或多或少都会影响下单服务的性能。还有一个问题是，如果 Redis 本身出现故障，写入数据失败，还会导致下单失败，等于是降低了下单服务性能和可用性，这样肯定不行。

**对于像订单服务这类核心的业务，一个可行的方法是，我们启动一个更新订单缓存的服务，接收订单变更的 MQ 消息，然后更新 Redis 中缓存的订单数据。**因为这类核心的业务数据，使用方非常多，本来就需要发消息，增加一个消费订阅基本没什么成本，订单服务本身也不需要做任何更改。



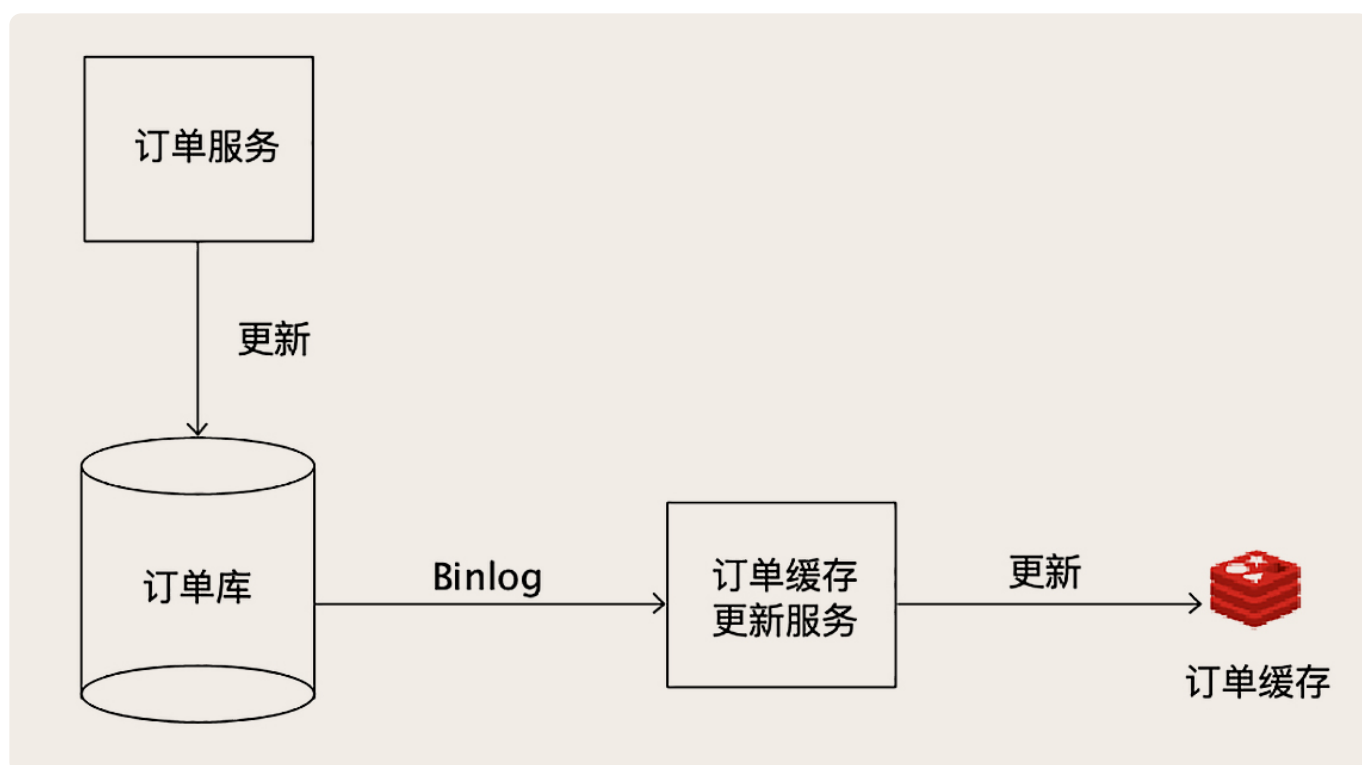
唯一需要担心的一个问题是，如果丢消息了怎么办？因为现在消息是缓存数据的唯一来源，一旦出现丢消息，缓存里缺失的那条数据永远不会被补上。所以，必须保证整个消息链条的可靠性，不过好在现在的 MQ 集群，比如像 Kafka 或者 RocketMQ，它都有高可用和高可靠的保证机制，只要你正确配置好，是可以满足数据可靠性要求的。

像订单服务这样，本来就有现成的数据变更消息可以订阅，这样更新缓存还是一个不错的选择，因为实现起来很简单，对系统的其他模块完全没有侵入。

## 使用 Binlog 实时更新 Redis 缓存

如果我们要缓存的数据，本来没有一份数据更新的 MQ 消息可以订阅怎么办？很多大厂都采用的，也是更通用的解决方案是这样的。

数据更新服务只负责处理业务逻辑，更新 MySQL，完全不用管如何去更新缓存。负责更新缓存的服务，把自己伪装成一个 MySQL 的从节点，从 MySQL 接收 Binlog，解析 Binlog 之后，可以得到实时的数据变更信息，然后根据这个变更信息去更新 Redis 缓存。



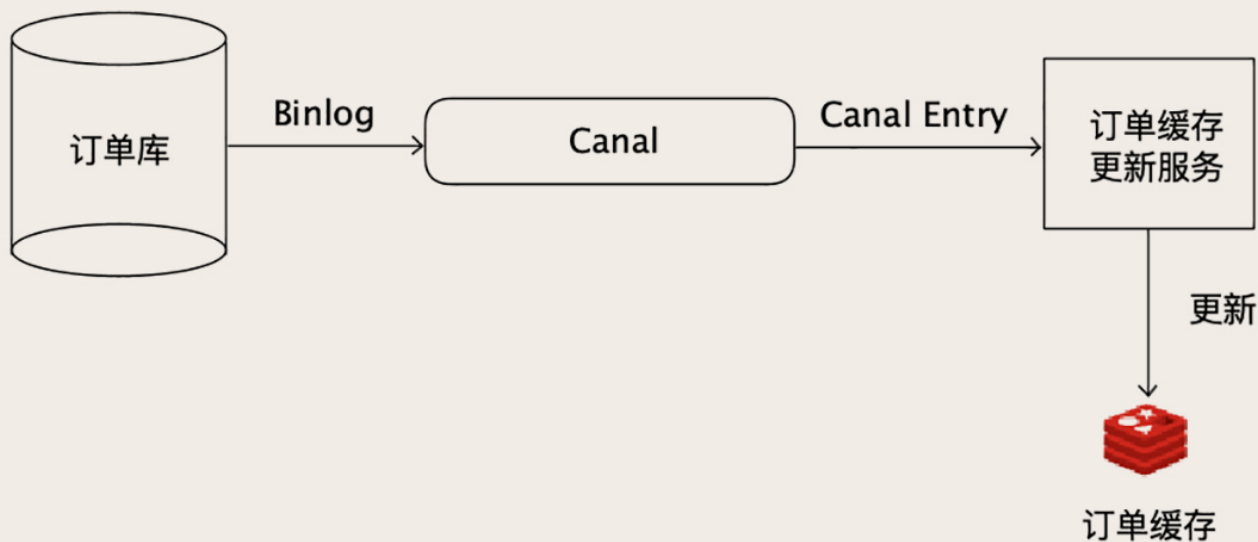
这种收 Binlog 更新缓存的方案，和刚刚我们讲到的，收 MQ 消息更新缓存的方案，其实它们的实现思路是一样的，都是异步订阅实时数据变更信息，去更新 Redis。只不过，直接读取 Binlog 这种方式，它的通用性更强。不要求订单服务再发订单消息了，订单更新服务也不用费劲去解决“发消息失败怎么办？”这种数据一致性问题了。

而且，在整个缓存更新链路上，减少了一个收发 MQ 的环节，从 MySQL 更新到 Redis 更新的时延更短，出现故障的可能性也更低，所以很多大厂更青睐于这种方案。

这个方案唯一的缺点是，实现订单缓存更新服务有点儿复杂，毕竟不像收消息，拿到的直接就是订单数据，解析 Binlog 还是挺麻烦的。

有很多开源的项目就提供了订阅和解析 MySQL Binlog 的功能，下面我们以比较常用的开源项目 [Canal](#) 为例，来演示一下如何实时接收 Binlog 更新 Redis 缓存。

Canal 模拟 MySQL 主从复制的交互协议，把自己伪装成一个 MySQL 的从节点，向 MySQL 主节点发送 dump 请求，MySQL 收到请求后，就会开始推送 Binlog 给 Canal，Canal 解析 Binlog 字节流之后，转换为便于读取的结构化数据，供下游程序订阅使用。下图是 Canal 的工作原理：



在我们这个示例中，MySQL 和 Redis 都运行在本地的默认端口上，MySQL 的端口为 3306，Redis 的端口为 6379。为了便于大家操作，我们还是以《[04 | 事务：账户余额总是对不上账，怎么办？](#)》这节课中的账户余额表 `account_balance` 作为演示数据。

首先，下载并解压 Canal 最新的 1.1.4 版本到本地：

复制代码

```
1 wget https://github.com/alibaba/canal/releases/download/canal-1.1.4/canal.deployer.tar.gz
2 tar zvf canal.deployer-1.1.4.tar.gz
```

然后来配置 MySQL，我们需要在 MySQL 的配置文件中开启 Binlog，并设置 Binlog 的格式为 ROW 格式。

复制代码

```
1 [mysqld]
2 log-bin=mysql-bin # 开启Binlog
3 binlog-format=ROW # 设置Binlog格式为ROW
4 server_id=1 # 配置一个ServerID
```

给 Canal 开一个专门的 MySQL 用户并授权，确保这个用户有复制 Binlog 的权限：

复制代码

```
1 CREATE USER canal IDENTIFIED BY 'canal';
```


```
2 GRANT SELECT, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'canal'@'%';
3 FLUSH PRIVILEGES;
```

重启一下 MySQL，确保所有的配置生效。重启后检查一下当前的 Binlog 文件和位置：

```
mysql> show master status;
```

File	Position	Binlog_Do_DB	Binlog_Ignore_DB	Executed_Gtid_Set
binlog.000009	155			

记录下 File 和 Position 两列的值，然后我们来配置 Canal。编辑 Canal 的实例配置文件 canal/conf/example/instance.properties，以便让 Canal 连接到我们的 MySQL 上。

 复制代码

```
1 canal.instance.gtidon=false
2
3
4 # position info
5 canal.instance.master.address=127.0.0.1:3306
6 canal.instance.master.journal.name=binlog.000009
7 canal.instance.master.position=155
8 canal.instance.master.timestamp=
9 canal.instance.master.gtid=
10
11
12 # username/password
13 canal.instance.dbUsername=canal
14 canal.instance.dbPassword=canal
15 canal.instance.connectionCharset = UTF-8
16 canal.instance.defaultDatabaseName=test
17 # table regex
18 canal.instance.filter.regex=.*\\..
```

这个配置文件需要配置 MySQL 的连接地址、库名、用户名和密码之外，还需要配置 canal.instance.master.journal.name 和 canal.instance.master.position 这两个属性，取值就是刚刚记录的 File 和 Position 两列。然后就可以启动 Canal 服务了：


 复制代码

```
1 canal/bin/startup.sh
```

启动之后看一下日志文件 canal/logs/example/example.log，如果里面没有报错，就说明启动成功并连接到我们的 MySQL 上了。

Canal 服务启动后，会开启一个端口（11111）等待客户端连接，客户端连接上 Canal 服务之后，可以从 Canal 服务拉取数据，每拉取一批数据，正确写入 Redis 之后，给 Canal 服务返回处理成功的响应。如果发生客户端程序宕机或者处理失败等异常情况，Canal 服务没收到处理成功的响应，下次客户端来拉取的还是同一批数据，这样就可以保证顺序并且不会丢数据。

接下来我们来开发账户余额缓存的更新程序，以下的代码都是用 Java 语言编写的：

 复制代码

```
1 while (true) {
2     Message message = connector.getWithoutAck(batchSize); // 获取指定数量的数据
3     long batchId = message.getId();
4     try {
5         int size = message.getEntries().size();
6         if (batchId == -1 || size == 0) {
7             Thread.sleep(1000);
8         } else {
9             processEntries(message.getEntries(), jedis);
10        }
11
12
13        connector.ack(batchId); // 提交确认
14    } catch (Throwable t) {
15        connector.rollback(batchId); // 处理失败，回滚数据
16    }
17 }
```

这个程序逻辑也不复杂，程序启动并连接到 Canal 服务后，就不停地拉数据，如果没有数据就睡一会儿，有数据就调用 processEntries 方法处理更新缓存。每批数据更新成功后，就调用 ack 方法给 Canal 服务返回成功响应，如果失败抛异常就回滚。下面是 processEntries 方法的主要代码：

 复制代码

```
1 for (CanalEntry.RowData rowData : rowChage.getRowDatasList()) {
2     if (eventType == CanalEntry.EventType.DELETE) { // 删除
```



```
3         jedis.del(row2Key("user_id", rowData.getBeforeColumnsList()));
4     } else if (eventType == CanalEntry.EventType.INSERT) { // 插入
5         jedis.set(row2Key("user_id", rowData.getAfterColumnsList()), row2Value
6     } else { // 更新
7         jedis.set(row2Key("user_id", rowData.getAfterColumnsList()), row2Value
8     }
9 }
```

这里面根据事件类型来分别处理，如果 MySQL 中的数据删除了，就删除 Redis 中对应的数据。如果是更新和插入操作，那就调用 Redis 的 SET 命令来写入数据。

把这个账户缓存更新服务启动后，我们来验证一下，我们在账户余额表插入一条记录：

```
1 mysql> insert into account_balance values (888, 100, NOW(), 999);
```

 复制代码

然后来看一下 Redis 缓存：

```
1 127.0.0.1:6379> get 888
2 "{ \"log_id\": \"999\", \"balance\": \"100\", \"user_id\": \"888\", \"timestamp\": \"2018-08-08 10:00:00\" }
```

 复制代码

可以看到数据已经自动同步到 Redis 中去了。我把这个示例的完整代码放在了 [GitHub](#) 上供你参考。

## 小结

在处理超大规模并发的场景时，由于并发请求的数量非常大，即使少量的缓存穿透，也有可能打死数据库引发雪崩效应。对于这种情况，我们可以缓存全量数据来彻底避免缓存穿透问题。

对于缓存数据更新的方法，可以订阅数据更新的 MQ 消息来异步更新缓存，更通用的方法是，把缓存更新服务伪装成一个 MySQL 的从节点，订阅 MySQL 的 Binlog，通过 Binlog 来更新 Redis 缓存。



需要特别注意的是，无论是用 MQ 还是 Canal 来异步更新缓存，对整个更新服务的数据可靠性和实时性要求都比较高，数据丢失或者更新慢了，都会造成 Redis 中的数据与 MySQL 中数据不同步。在把这套方案应用到生产环境中去的时候，需要考虑一旦出现不同步问题时的降级或补偿方案。

## 思考题

课后请你思考一下，如果出现缓存不同步的情况，在你负责的业务场景下，该如何降级或者补偿？欢迎你在留言区与我讨论。

感谢你的阅读，如果你觉得今天的内容对你有帮助，也欢迎把它分享给你的朋友。

# 后端存储实战课

## 类电商平台存储技术应用指南

李玥

京东零售计算存储平台部资深架构师



新版升级：点击「🔗 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 16 | 用Redis构建缓存集群的最佳实践有哪些？

下一篇 18 | 分布式存储：你知道对象存储是如何保存图片文件的吗？

## 精选留言 (13)

写留言



李玥 置顶



2020-04-07

Hi, 我是李玥。

这里回顾一下上节课的思考题：

课后请你再去看看 HDFS，它在解决分片、复制和高可用这几方面，哪些是“抄作...  
展开 ▾



每天晒白牙

2020-04-04

我们的系统也采用了 canal 监听 binlog 变更来异步更新 ES 和 redis 中的数据的方式

不过我们的方案多了3个步骤

- 1.canal 把消息发到 kafka 中，应用程序监听 topic
- 2.应用程序收到消息后，根据 id 重新读 mysql...

展开 ▾



芒果

2020-04-04

思考题我的想法是：

- 1.如果缓存存在不同步的情况，那么客户端的数据就不是最新数据。如果用户不能接受数据不同步（比如：刚刚下的订单但是购物记录里面没有），作为用户一般都会进行手动刷新，服务端接收到用户手动刷新的请求时，直接去查数据库，然后通过老师之前讲的cache aside pattern的方式更新缓存。...

展开 ▾



Dovelol

2020-04-04

老师好，用binlog方式同步mysql数据到redis，如果是已经在线运行很久的表数据，也适合转到这个方案吗？需要把之前的数据全部同步到redis中，重要的是该从binlog中的哪个位置开始呢。

作者回复: 这种情况需要先做一次全量同步，之后再开启binlog做增量同步。



正在减肥的胖籽。



2020-04-07

数据同步到redis中，但是redis突然down丢失一些数据，或者redis也会设置缓存时间。那么还是会打到库中？老师你们这里的方案是？或者说你们的redis过期时间怎么平衡？



飞翔

2020-04-07

老师 canal 是不是也做的集群 防止它当机了 redis 不同步了



CJJ

2020-04-06

老师，我还有个疑问。用mq去更新缓存数据，如若上面所说Redis出现故障，这应如何处理？我想到的是重试机制，但超过次数应当如何处理？

展开

作者回复: MQ消费的时候有自动重试机制，并且不建议这个地方加重试次数的限制。如果Redis故障，就让同步卡在那儿，等Redis恢复之后，就可以继续同步，这样不会丢数据。



Mq

2020-04-06

- 1.可以增加一个对账功能，对数据库跟缓存的数据，数据最好有个版本号或时间，把不一致把数据库的数据刷进缓存
- 2.提供刷用户缓存的服务，对投诉用户可以优先刷下



CJJ

2020-04-05

全量数据缓存，缓存同步有个时间差，请问老师这该如何处理？

作者回复: 就行MySQL主从同步时延一样，只能接受它。一般这个时延都是毫米级的，不会对业务有很大影响。



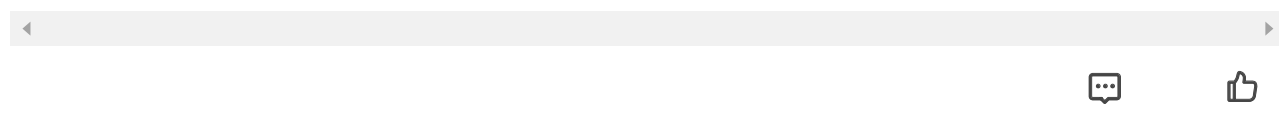
一剑

2020-04-05

这里有个问题，就是我们一般是把计算结果缓存到redis，但是基于日志的同步方式是直接同步了原始表数据，这中间是不是少了一环？

展开 ▾

作者回复: 这里面需要注意一下，Binlog中记录的是“数据变化”，而不仅仅是数据。



川杰

2020-04-04

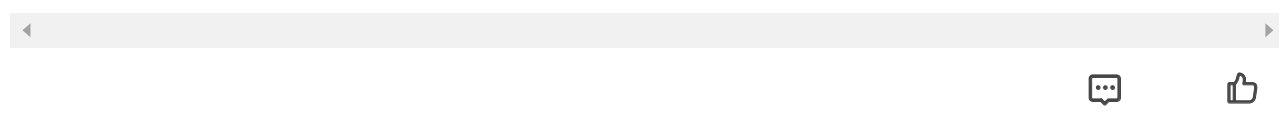
老师好，想问一个redis很基础的问题。

假设我们要对交易数据进行缓存。后端调用时，既有根据交易编号查找单个对象的方法，又有查询批量交易的方法。那我该怎么缓存交易数据呢？

利用key-value的方式可以解决根据交易编号查找的情况。那批量查询怎么处理？用队列吗？如果用队列，那岂不是一个交易数据要缓存两遍？（一个是队列，一个是key-value...

展开 ▾

作者回复: 一般批量查询的时候可以用Redis的集合数据结构，比如SET，SET中的Value可以保存交易编号，而不用保存交易数据。



leslie

2020-04-04

这其中其实有个坑：redis与mysql之间的设置，多少合适？太小访问太频繁，太大更新不及时。

电商中哪些业务的刷新频率的设置这个确实。。。

展开 ▾



饭团

2020-04-04

学到了，哈哈！之前一直看有解析binlog日志的做法！

展开 ▾



