支付系统设计：对账处理（二）

可以说，对账是支付系统最头疼的事情。每一笔交易，都要做到各参与者的记录能够吻合，没有偏差。对账系统的工作，是发现有差异的记录，即轧帐；然后通过人工或者自动的方式，解决这些差异，即平帐。

对电商系统来说，每一笔交易，在所有相关主体侧都要能对得上：

交易主体，如果发起人是个人，必须能够从个人交易历史记录中找到这笔交易。但大部分人不会保留电子记录，所以一般是提供可以下载的账单或交易记录，让用户自己对去。
交易对手，一般是商户。商户侧对账处理同用户侧，也仅仅提供对账单。
交易渠道侧，这是对账的重点，一是核实交易流水，二是核实交易佣金，毕竟是租用人家通道做结算的。

那有哪些记录需要对账？ 目前主要是两个：一个是交易记录；一个是退款记录。

对账处理流程

一般来说，对账流程涉及到如下步骤： 渠道对账单下载、本地交易记录准备、轧账、平账。

渠道对账单下载

银行，第三方支付，银联等，基本都会提供对账单下载的功能。不过也有少数工作做不到位或者太到位的银行，只提供账单查询后台，不提供对账单下载功能。

对开发人员来说，这里有几个坑：

对账单格式不一。文本，XML，csv 的都有。为了后续能够统一处理，在账单下载完成后，需要进行标准化处理。
下载方式不一，HTTP，HTTPS，FTP 的，都有。下载程序需要按照渠道的协议来处理。
下载时间不一，一般是凌晨 1 点后，到中午 12 才能用的也有。如果在预定的时间取不到数据，需要注意重试读取。

稳定性差。FTP 服务器出问题那是常有的事。渠道侧解决方案往往就是重启。所以重试机制是必要的。

看一下第三方支付的对账单情况：

渠道	对账周期	账单提供方式	账单文件格式
支付宝	每天 2:10	HTTPS	XML
支付宝退款	每天3:10	HTTPS	XML
百付宝	每天7:00	FTP	TXT
百付宝退款	每天7:00	FTP	TXT
微信支付	每天10:30	HTTPS	TXT
微信退款	每天10:30	HTTPS	TXT

银行直连的对账情况：

银行	对账形式	对账周期	打款周期
交行	接口/商户对账系统	日对账	日结（T+1）
建行	接口	日对账	日结（T+1）
工行	登录网银的方式手动下载	日对账	日结（T+1）
浦发	信用卡登录自助平台获取对账文件， 借记卡通过接口形式提供对账文件	日对账	日结（T+1）
农行	银行定时推送对账文件	日对账	日结（T+0）
中行	银行定时推送对账文件	日对账	日结（T+1）
招行	银行定时推送对账文件	日对账	日结（T+1）

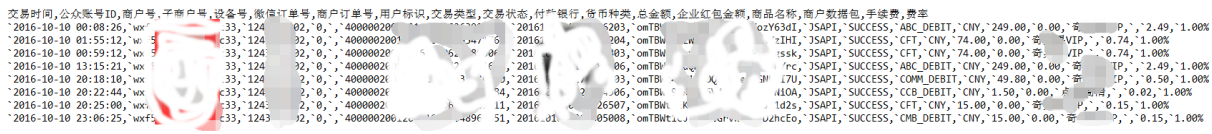
技术选型上，HTTP(S)用 apache httpclient 即可实现链接池和断点续传， FTP 也可以使用 Apache Commons Net API。 但不管是哪一个，都需要设置重试次数和链接超时间。重试次数和间隔的设置需要小心，重试太频繁，容易把服务器打死.；时间间隔太大，又会阻塞后续处理步骤。5～10 分钟是一个合适的重试间隔区间。

链接超时指在服务器出现问题时，连接在指定时间内获取不到数据即自动断开。这个很容易被忽略。我们有一次系统出问题，是渠道侧的 FTP 假死后重启，导致我们的客户端挂住，一直在等待重新链接。

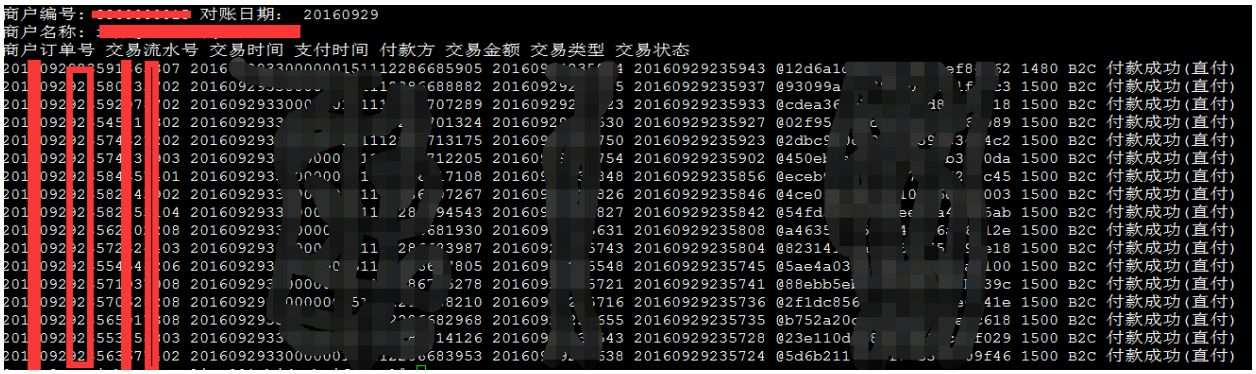
渠道对账单标准化

找个例子大家看看， 比如微信的对账单，他是 csv 格式的，包括如下信息：

交易时间：这是在微信侧的支付完成的时间。 这个时间会成为一个陷阱。
公众账号 ID，商户号, 子商户号, 设备号： 这些信息需要做验证，确保是自己的单子，不要让微信把老王家的单子也给发过来了；
微信订单号, 商户订单号： 这两个是对单的核心。前者是微信侧产生的订单号，在微信支付接口返回值中有。但是万一收不到这个返回值，那在本地记录中可能就空了。 后者是我们发送给微信的订单号，一般用这个来做对单依据。两边的数据中都会有这个值。
用户标识, 交易类型, 交易状态, 付款银行, 货币种类, 总金额, 企业红包金额： 这几个就是对单的核心字段，必须确保双方是一致的。
商品名称, 商户数据包, 手续费, 费率：这些是可选验证。



而某宝的对账单，是文本格式的，用空格隔开。他们家的就简单很多，只有商户订单号，交易流水号，交易时间，支付时间，付款方，交易金额，交易类型，交易状态这些字段。



由于每个渠道的账单格式都不尽相同， 在得到账单后，下一步是对账单做标准化处理，这样轧帐以及后续工作就可以统一处理了。 标准化后的账单数据可以放在文件系统或者数据库中。这取决于交易数据量。每天百万以上的量，还是使用文件系统，比较合适。数据库操作相对比较慢，也浪费资源。

基于文件系统的标准化涉及如下内容：

文件格式标准化：统一使用 csv 或者 json 或者 xml 格式。如果是使用 hadoop 或者 spark 来对账，使用 csv 是个不错的选择。
文件存储统一化：文件目录，文件名都需要遵循统一命名规范。

为了加快处理速度，我们使用 hdfs 作为文件系统，有利于后续的对账的处理。

本地交易记录准备

本地交易记录的准备，总的来说有如下方法： - 啥都不做，直接用原始数据。鉴于大部分系统使用的是 mysql，这也意味着在 MySQL 上做对账。对账时需要大量的数据查找工作，必然会影响线上业务。在数据规模较大，比如超过 100 万时，就不太合适了。

当然，还有一个选择是使用备库来执行对账，这样既简单，也不影响线上业务。这是典型的空间换时间的做法。
如果业务大到需要分表分库才能处理，那对账数据准备也不一样。使用分库也不现实，因为分库一般是按照主体 id，而不是渠道 id，来分库，这样对账就需要在多个库上进行，效率反而降低了。而对分表分库建立从库也非常耗费资源。这种情况下，需要同步一份数据到(hdfs)文件系统中，或者 NOSQL 数据库上。

由于交易记录是支付系统核心数据，有大量的应用，如信用、风控等，都需要交易记录数据。这些应用对交易记录的需求还不完全一致，为了提升性能， 交易记录会使用异步的方式来将数据投递给使用方。 交易记录在入库时，投递消息到消息系统中。使用方监听这个消息，一旦收到新消息，则从交易记录库中查询数据，获取数据并更新到库中。关于此类数据同步的文章不少，这里就不详细介绍。

轧帐

轧帐是按照客户订单号来比较本地交易记录和渠道交易记录是否一致。从算法角度，是计算两个数组的差异。在单机运行时，可以采用的算法不少，这里不详细介绍。 我们推荐采用 mapreduce 来轧帐，这有个优势，可以按照订单号将渠道提供的记录和本地记录 shuffle 到同一个 reduce 处理上，这样就可以很容易进行数据比对。 轧帐中最大的坑，莫过于切分点的问题。

比如以整 0 点为切分点，那存在一个问题，本地 23:59 发起的交易，到了渠道侧，可能会在 00:01 处理，这一笔交易变成第二天的帐了。实际处理中，一笔交易在渠道侧处理，花上几分钟都有可能。 对于切分点附近无法确认的帐，做一个时间窗，在时间窗内的数据，留待第二天对账时继续处理。

平帐

发现两边不一致的数据，那应该如何处理？数据量不大时，记录起来，人工甄别就行。但如果数据量很大，每天上千条，人工处理就成本太高了。这个没有统一的处理方法，需要根据有问题的数据，做个分析，然后做自动处理。 针对交易记录的对账的处理，主要有如下情况：

本地未支付，支付渠道已支付。这主要是本地未正确接收到渠道下发的异步通知导致。
一般处理是将本地状态修改为已支付，并做响应的后续处理，比如通知业务方等。
本地已支付，支付渠道已支付，但是金额不同，这个需要人工核查。
本地已支付，但是支付渠道中无记录；或者本地无记录，支付渠道有记录。在排除跨日因素外，这种情况非常少见，需要了解具体原因后做处理。

针对退款的对账处理，主要有如下情况：

本地未退款，支付渠道已退款，则以支付渠道为准，修改本地为已退款状态，并触发后续处理。
本地已退款、支付渠道已退款，但是金额不同，需要人工核查；
本地已退款，但是支付渠道无记录；或者支付渠道有记录，但是本地没有。 在排除跨日因素外， 这种情况非常少见，需要了解具体原因后做处理。

总之，对账工作，即复杂也不复杂。需要细心，对业务要有深入的了解，并选择合适的架构。

相关阅读

支付系统设计：支付系统的账户模型（一）

作者：凤凰牌老熊，程序员 & 架构师，来自中科大的本科，研究生在软件所学习。先后在中科辅龙、三星（中国）研究院和国内一些大型的互联网公司呆过。在中科辅龙公司负

责电子政务内容管理系统建设，负责研发龙驭系列产品的研发，这款产品最终实施到 2000 多个电子政务网站上，期间也参与了一些支付反洗钱以及支付系统的建设。之后在三星中国研究院，负责自然语言处理（NLP）以及智能家居相关项目。智能家居项目在 2014CES 消费电子展上作为三星重点项目推介。2014 年开始加入爱奇艺公司，负责数据仓库和支付系统的建设。