资金之旅——走进账务清算域（三）：账务篇下

寇恂 2016-07-05 12:40:19

微账务 稳定性 账务 资金 高可用 修改标签

上一篇： 资金之旅——走进账务清算域（二）：账务篇上

账务平台的技术挑战

账务平台各系统，尤其是主账务系统acctrans，处在交易支付主链路上，是账务清算域中高并发、高容量、高可用要求最高的系统。由于这个系统是直接操作客户资金的，哪怕放在全站，也算是技术挑战最高的业务系统之一。

自诞生以来，尤其是2010年开始支撑大促以来，账务平台在“三高”方面积累了丰富的经验，下面选取几个**有代表性的课题**做一下分享。由于这些课题每一个都可以展开很多，而且很多课题在ATIT、DOC上我们都曾做过专题分享，部分也有授课视频。因此限于篇幅，本文就不做过多着墨，只是做一些关键点的提炼。

热点账户

所有存在一定交易量的账务系统都会面临热点账户的问题，其根源来自于数据库的悲观锁。目前账务的解决方案 有两种：缓冲记账和日终汇总记账，一般业界也是类似的方案。

缓冲补账，采用“削峰填谷”的思想，对于热点账户，账务在预处理和提交过程中，不进行实时余额的锁定检查和更新，只记录记账凭证。然后定时通过分布式调度任务，按照账户的维度进行汇总更新余额，逐一补记账务明细，发送账务变动消息。
缓冲补账的关键点在于补账的时间片拆分任务的处理量，既要每个账户都能补到，又要保证每次补账的明确周期，不要上个批次还没补完，下个批次又来了，进而引发雪崩。同时还要确保数据库的压力在可控范围之内。目前这些参数都已支持动态调整实时配置生效。

由于缓冲补账不实施校验余额，故存在账户透支风险。所以绝大多数缓冲补账都是使用在收款缓冲场景。付款缓冲一般用在一些内部账户，或者可以确保风险的外部账户如彩票发奖账户上面。

缓冲记账在实际应用中的一个难点在于，不是每个热点账户都是能够提前识别提前配置的。尤其是一些商户账户，由于突然搞活动等原因，造成交易额突然猛增，变成热点，大量占用数据库连接，降低整体集群的吞吐量。为了应对这种场景，账务平台借助监控平台、弹性计算平台和分布式资源管理平台的能力，通过监控特殊的日志，自动驱动配置缓冲补账。这种方式使得热点账户应急的时间基本上降到了1分钟级别，极大地提升了业务的可用性。
对于一些交易量巨大，连缓冲补账都无法搞定的账户，比如担保交易、天猫佣金、移动话费等，账务采取汇总记账的方式，逐笔记录记账凭证，日终汇总一次更新余额。和缓冲补账不同的是，汇总记账不再逐笔补记明细和发送动账通知。为了保证记账凭证不都落在汇总账号所在的分表中，这儿有个特殊处理，将这些凭证按照对手方账号的维度进行数据路由离散。

数据库拆分

数据库拆分，如今早已成为提升系统容量，杜绝资源单点的一个常规手段，其技术本身并没有太多神秘。在支付宝，数据源的管理和路由已经被TDDL/ZDAL封装好，对于应用，更多的是需要结合自身的业务特性考虑分库的策略。主账务核心[illegible]拆分、水平拆分、混合拆分，过程很有代表性。

首先，2010年淘宝商城举行第一次真正意义上的双11大促，账务所在的payplus库在当天已经被压到极致，最后靠着紧急杀死同在payplus库上的会计系统才涉险过关。因此在11年，必须要进行拆库。当时采取了相对比较保守的垂直拆库，将账务的交易数据拆分到新的账务前置库（payfront），账户信息和账务明细仍然保留在payplus库上，两个库的压力占比大概在3:7的样子，靠着这个架构，我们撑过了2011年大促。

随后，我们我们启动了“账三丰”技改项目，对账务核心进行了三项大改，极大提升了容量和稳定性。其中两项就是对已经独立的前置库进行水平拆分，同时将原来实时写入到payplus库的账务明细数据，改成异步的方式，写到新的acclog库中。再往后，我们对payplus库的账户信息数据和acclog库的账务明细数据进行了彻底的水平拆分，账务核心的数据库架构逐步演化成了如下模式（数据库名字不用关注，后面在不停地变化）：

Failover

关于Failover，想必大家也早已不陌生。对于基于流水型数据的系统的Failover相对好做。由于流水之间无关联性，数据库宕机，只要拉起一个新的数据库，后续创建的业务就可以正常执行。且经过交易、支付的Failover项目先行探路，账务也很快完成了账务前置库（账务交易数据，流水型）的Failover改造。过程中重点考虑的点，主要在于切换、回切过程中针对自身业务的开关控制和兼容性处理。

接下来该轮到账号库了。总所周知，一个账户的余额，可能有无限多种情况。这是一种状态型数据，账务的Failover，也就是无穷状态机系统的Failover。账户的当前余额与前序状态（上次余额）息息相关，稍有不慎，余额取得不精准，就是资金风险，这是一个相当大的难题和挑战。我们专门成立了一个账务难题小组，汇聚了账务、中间件、DBA、运维等各个团队的大牛，经过无数次脑暴、讨论和推演，最终决定采取以下“两步走”的方案。

第一步：网关型Failover。经过对用户资金状态和支付习惯的分析，我们发现大概有70%的用户支付行为，都是使用银行卡而不是用户余额进行支付的。也就是说，大多数用户在支付时，其实并不依赖于他的余额。所以，考虑到当时账号库已经拆成20个，当其中某个库宕掉后，只要我们能保证用户依然能够使用银行卡支付，那么理论上就可以使得对交易的影响降低到1/20*30%=1.5%（实际值考虑到数据交叉，实际影响比例会高一些，但基本在5%以下），影响非常小。

网关型Failover的核心思想就是先建立读库。读库中的账户余额虽然有1秒左右的延迟，但它们只用于查询，对业务基本没有影响。建立配套的Failover库，当切换到Failover模式下，用户支付的账务操作都发生在failover库，不依赖于余额数据。当用户用银行卡支付时，先在Failover库做一笔充值，然后推进支付处理。理想情况下Failover库的余额是0。当主库恢复后，如果Failover库中用户有余额，则进行merge，和主库中事故前的余额进行合并。过程如下图所示。

第二步：完整型Failover。完整型Failover在网关型Failover的基础上有了进一步改进，将故障发生前一小段时间因资金变动导致读库余额尚未同步完成的账号纳入黑名单，将业务影响降到了最低。其中的关键点在于利用C-ZONE的数据读取能力，每一次账户余额变更后，通过消息将最新余额发到C-ZONE写入tair，与读库结合后可以提供准确的最新余额。如果在数据库宕机后仍有尚未投递成功的消息，表示这部分消息的余额可能不准确，账务系统将在Failover期间拒绝这些账号的记账处理。说明图如下：

账务完整型Failover最先在余额宝账务上实现，后推广到主账务、用信账务等。用创造性的思路，巧妙而彻底地解决了状态型数据的Failover难题。关于账务Failover的详细介绍，可以参看“扩展阅读”环节。

账务单元化改造（LDC）

全站单元化的思路是根据用户账号的维度（uid的倒数2,3位）进行服务和数据的拆分，使得一个用户的一次完整业务行为，均发生在同一个逻辑单元（R-ZONE）中。基于支付宝的SOA架构，服务本身是无状态的，配合相应的流量调配组件和中间件，只需要作一点适配就可以做到单元化。相对比较困难的是数据的单元化。

账务的数据既有流水型数据（账务交易信息），也有状态型数据（账户信息）。前者可以根据买家的uid进行单元化路由，而后者，一次账务操作很可能涉及到收付双方两个账户，而这两个账户又有很大可能在不同的逻辑单元中，势必会出现跨ZONE调用的场景。跨ZONE调用分为服务跨ZONE和SQL跨ZONE。一次记账处理会涉及到多次对账户相关信息的SQL操作，时延不可接受，因此关键点在于针对每个账户单边记账处理的服务化改造。将每次记账操作的跨ZONE调用减少

除此之外，还涉及到诸如查询迁移至C-ZONE，用户的理财基金账户、余额宝集分宝等微账务数据迁移到与主账户数据同一逻辑单元（基金理财本ZONE化、微账务本ZONE化）等配套改造项目，详细介绍可以参看“扩展阅读”环节。

单元化后，结合已有Failover能力，账务系统基本具备了异地多活的容灾能力。相应的架构如下：

弹性化

弹性化是指应用计算资源可以根据容量需要，灵活扩容或缩容，以到达按需使用，节省成本的目的。在云计算技术十分成熟的今天，弹性化是一项非常有价值的能力。

回到我们的场景，双十一大促/新春红包等活动，是典型的对计算资源高要求的场景。如果我们的服务器按照双十一的容量进行采购，那么到了平时，服务器又十分空闲，造成极大的浪费。

对于账务交易数据，是无状态的流水型数据，数据间基本没有关联，生命周期短。无论我们将它写在哪个数据库中，只要在短时间内完成记账处理，就将不会对后续业务造成任何影响。因此我们可以在高峰期，将它们写到临时调配的数据库中，分散正式库的压力；到业务低峰期，再将临时数据库归还。这儿的“临时调用的数据库”，其实就是平时闲置的Failover库，也可以是云计算的数据库。
账务平台将这个思想在2015年大促以及2016年新春红包中实施，取得了相当大的成功。这个思想也将在2016年推广到全站，成为应对大促的一项“标配”。

账务高可用组件（acchc）

账务在不断解决难题，不断迎接新挑战的基础上，也期望能通过自身这么多年高可用方面的积累，进行对外输出。从2015年开始，账务平台将一些主要的技术方案沉淀成公共组件——账务高可用组件（acchc），利用该组件，其他系统可以比较简单的进行能力复用。例如完整版failover方案，已经通过该组件在微账务上一些重要子账务如用信账务上得到使用。目前acchc上主要集成了三项能力：容灾（只读failover/网关failover/完整failover/明细failover）、弹性化、SLA限流。

账务技术扩展阅读

账务failover DB设计（by羽磬）：http://atit.alipay.net/index.php?r=blog/detail&qid=1906
账务Failover系列（by许由）：
概述：http://atit.alipay.net/index.php?r=blog/detail&qid=1892
选型：http://atit.alipay.net/index.php?r=blog/detail&qid=1911
余额宝账实践：http://atit.alipay.net/index.php?r=blog/detail&qid=1957
架构域接入LDC的改造思路（by祢衡）http://atit.alipay.net/index.php?r=blog/detail&qid=2241
账务LDC-微账务本zone化（by极天）http://atit.alipay.net/index.php?r=blog/detail&qid=2243
账务LDC-余额宝收益本zone化（by极天）http://atit.alipay.net/index.php?r=blog/detail&qid=2249
账务LDC——主账务单边记账服务化改造（by静俭）http://atit.alipay.net/index.php?r=blog/detail&qid=2255
双十一容量提升：账务数据弹性化设计（by静俭）http://atit.alipay.net/index.php?r=blog/detail&qid=3072

下一篇：资金之旅——走进账务清算域（四）：核算篇

评论文章(3)

文章点评

评论