Computer Science and Application 计算机科学与应用, 2019, 9(7), 1334-1341
Published Online July 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.97150

Hans汉斯

Research on Distributed Accounting Model of Centralized Account in Payment System

Peng Xiao, Wanmin Cui, Cuiping Li, Wanjing Jing

Yinqing Technology (Beijing) Co. Ltd., Beijing
Email: pengxiao@cncc.cn, wmcui@cncc.cn, cpli@cncc.cn, wjjing@cncc.cn

Received: Jul. 1st, 2019; accepted: Jul. 15th, 2019; published: Jul. 22nd, 2019

Abstract

In Payment System, the high traffic volume may lead to the problem that the single settlement account becomes a hot account. At the same time, to further solve the distributed architecture transformation of settlement account system under the centralized account scenario, this paper studies the feasibility of the distributed accounting model in Payment System from two modes: debit and credit separation and shadow account, and gives a comparison and applicable scenarios of the two modes in Payment System at the end.

Keywords

Payment System, Hot Account, Centralized Account, Distributed Accounting Model

支付系统集中账户分布式记账模型研究

肖　鹏，崔婉旻，李翠平，景婉婧

银清科技(北京)有限公司，北京
Email: pengxiao@cncc.cn, wmcui@cncc.cn, cpli@cncc.cn, wjjing@cncc.cn

收稿日期：2019年7月1日；录用日期：2019年7月15日；发布日期：2019年7月22日

摘　要

针对银行间支付系统高峰时业务量高有可能导致单一清算账户成为热点账户的问题，同时为进一步解决集中账户场景下的支付清算系统分布式架构改造，本文从借贷分离和影子账户两种模式研究分布式记账模型在支付系统中的可行性并在结尾给出了两种模式的对比及在支付系统中的适用场景。

文章引用：肖鹏，崔婉旻，李翠平，景婉婧．支付系统集中账户分布式记账模型研究[J]．计算机科学与应用, 2019, 9(7): 1334-1341. DOI: 10.12677/csa.2019.97150

关键词

支付系统，热点账户，集中账户，分布式记账模型

1. 引言

1.1. 分布式记账模型研究背景

支付系统为适应商业银行数据大集中趋势的需求，其接入方式由多点接入、多点清算过渡为一点接入、一点清算，目前能支持的峰值每小时 50 万次的单一账户访问量，超过商业银行峰值的最大业务量，能够满足当前的业务处理需求。但是随着我国社会经济快速发展，经济规模不断扩大，交易活动日益频繁，在可预见的未来，大型商业银行在高峰时业务量有可能突破每小时 50 万笔，导致单一清算账户成为热点账户。

目前支付系统更多的采用集中式架构，其特点是架构简捷稳定，便于纵向扩展与管理，技术成熟，但是集中式架构对高端设备依赖较强，业务连续性指标有待加强。

为进一步解决集中账户场景下的支付清算系统分布式架构改造，提出构建支付系统分布式记账模型研究。从借贷分离和影子账户两种模式研究其在支付系统中的可行性。

1.2. 分布式记账模型研究目标

1) 支持应用分布式改造

应用采用分布式架构，实现横向可扩展性。

2) 解决热点账户问题

3) 提高业务连续性

解决当前集中式架构依赖高端软硬件设备、系统健壮性弱等问题，部分节点故障时，保障业务连续性不受影响。

2. 分布式记账模型研究

2.1. 借贷分离

将支付业务拆分成借、贷两个事务，实现柔性事务，确保最终一致性。多实例架构下，各实例分别负责部分账户的记账处理。

借贷分离的好处是一旦发生区域性故障，可以降低故障的影响面，只有部分行的账户处理受到影响。

2.1.1. 基于 TCC 的补偿性事务原理

一个完整的 TCC 事务参与方包括三部分[1]，如图 1 所示：

1) 主业务服务

主业务服务为整个业务活动的发起方，如贷记业务商业银行 A 转账给商业银行 B，支付系统即是主业务服务。

2) 从业务服务

从业务服务负责提供 TCC 业务操作，是整个业务活动的操作方。从业务服务必须实现 Try、Confirm 和 Cancel 三个接口，供主业务服务调用。由于 Confirm 和 Cancel 操作可能被重复调用，故要求 Confirm 和 Cancel 两个接口必须是幂等的。前面的贷记业务场景中的商业银行 A 扣款和商业银行 B 增款即为从业务服务。

3) 业务活动管理器

业务活动管理器管理控制整个业务活动，包括记录维护 TCC 全局事务的事务状态和每个从业务服务的子事务状态，并在业务活动提交时确认所有的 TCC 型操作的 confirm 操作，在业务活动取消时调用所有 TCC 型操作的 cancel 操作。

TCC 把事务运行过程分成 Try、Confirm/Cancel 两个阶段，在每个阶段的逻辑由业务代码控制。其实际上是把数据库层的二阶段提交放在应用层来实现，对于数据库来说是一阶段提交，规避了数据库层 2PC 大颗粒度资源锁定导致的性能低下问题。

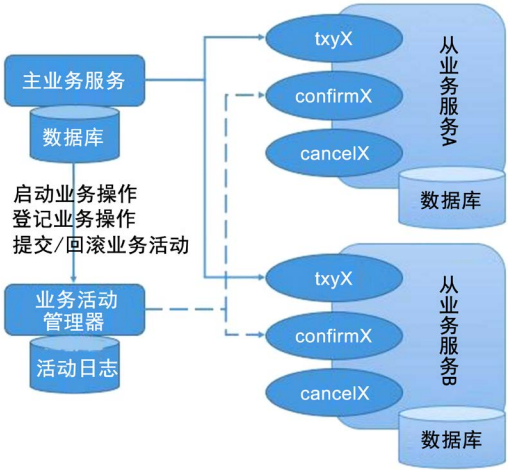

Figure 1. TCC transaction processing
图 1. TCC 事务处理

2.1.2. 支付清算业务 TCC 业务设计方案

支付业务清算 TCC 接口设计，如图 2 所示：

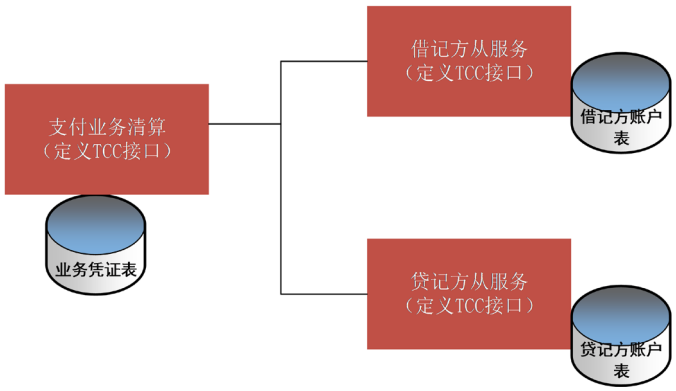

Figure 2. TCC interface design for payment business clearing
图 2. 支付业务清算 TCC 接口设计

Try：对支付清算报文进行报文格式、业务合法性、重账检查，通过检查则记录该业务凭证。调用借记方从服务的 Try 接口、贷记方从服务的 Try 接口，任何一个 Try 接口返回异常，都触发支付业务清算 Cancel 接口；

Confirm：更新业务凭证表业务处理状态为已清算；

Cancel：根据具体的情况做相应的操作，如头寸不足，更新业务凭证表业务处理状态为已排队；

借记方从服务 TCC 接口设计：

Try：检查借记行可用头寸是否足以支付，如果足够则对借记行账户进行资金圈存/直接扣减并提交(可根据业务需求做详细设计)。如果 Try 接口返回异常，调用 Cancel 接口；如果 Try 接口检测资源足够，调用 Confirm 接口。

Confirm：圈存资金改余额扣减/不做任何操作(对应 Try 中设计的具体操作，设计 Confirm 接口)

Cancel：圈存资金清 0/被减掉的加回来。(根据 Try 中设计的具体操作，进行相应的回滚操作)

贷记方从服务 TCC 接口设计：

Try：相关的业务检查。(可根据业务需求做详细设计，比如可直接在 Try 接口中将贷记方账户余额做增加操作。)

Confirm：贷记方账户余额做增加操作。(对应 Try 中设计的具体操作，设计 Confirm 接口)

Cancel：不做操作。(根据 Try 中设计的具体操作，进行相应的回滚操作，如果 Try 接口中设计了贷记方账户余额做增加操作，则 cancel 中进行相应的账户减操作)

2.2. 影子账户

影子账户是为流动性充裕的银行拆分账户，即内部通过分账户为各子账户单独建设跨地域的运行实例，并行受理业务，实现多中心多实例的运行架构。

通过对生产数据的统计查询分析，判断一个商业银行账户是否适合拆分子账户以及如何拆分子账户。

进行子账户拆分会产生分布式事务，分布式事务性能不佳[2]。因此，在应用设计上应结合实际业务特征，尽可能降低分布式事务发生几率。比如对于一些支付业务来往密切的银行可以划分为同一组行号集，使其尽量少的产生分布式事务。

拆分子账户的好处是对外暴露的是一个独立账户处理，对参与者无感知，也可以缓解热点账户问题。但是这种方式只适用于流动性充足的情况下。

2.2.1. 设计思路

本设计遵循对参与者透明，维持清算处理对参与者外特性不变这一基本设计原则。

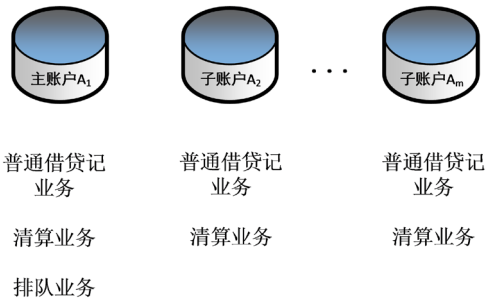

Figure 3. Account splitting and service acceptance
图 3. 账户拆分和业务受理

如图 3 所示，将可拆分的商业银行账户拆分成主账户(分配大部分的可用头寸)和多个子账户(分配部

分可用头寸)，主账户除受理普通的借贷记业务、清算业务外，还受理与排队业务相关的处理。对于不可拆分的商业银行账户，直接将其放在主账户实例中，与该行相关的业务路由到主账户实例处理。

当普通借贷记业务、清算业务路由到任一子账户实例(借贷双方均在同一个账户实例中)，头寸不足时，将其路由到主账户处理，当主账户头寸仍旧不足时，则进行头寸调拨，将子账户的头寸调拨至主账户，主账户头寸仍旧不足时，进行排队处理(达到触发金额起点的触发自动质押融资等服务。)；其他子账户发生借记业务，头寸增加时，在该行存在排队业务时，将子账户头寸调拨至主账户，用于解救排队业务；其他子账户发生借记业务，头寸增加时，在该行处于余额预警状态时，将子账户头寸调拨至主账户，用于余额预警解除。

该方案对于日间出现的头寸不足，需要进行头寸调拨，跨实例间交互，有一定的效率影响；对于清算窗口或日终业务对效率要求不高，可以解决分账户遇到的头寸不足判断的问题及部分流动性功能管理，头寸不足时，通过调拨来完成(资金池管理、日终拆借、风险预警与监控、圈存资金调整)。

2.2.2. 业务处理流程

业务处理流程如图 4 所示，

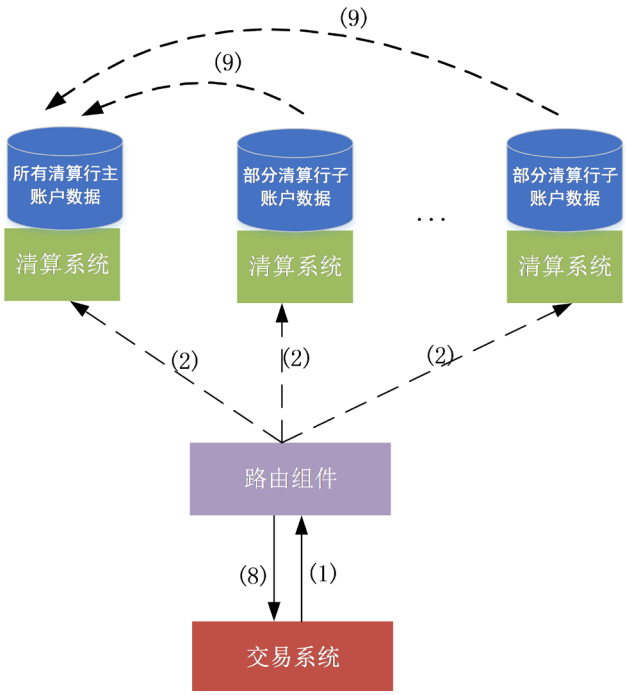

Figure 4. Business process
图 4. 业务处理流程

1) 交易系统将清算报文发送至路由组件系统；
2) 路由组件系统根据路由策略将报文分发至某一个清算系统实例中进行处理；
3) 清算系统对清算报文进行报文格式、业务合法性、重账检查，通过检查则记录该业务凭证；
4) 如果清算系统中借记行账户可用头寸足以支付，继续下一步，否则转 9)；
5) 清算系统针对该大额清算业务涉及的借记行、贷记行，进行账务处理；
6) 余额预警，仅监控主账户余额，
a) 如果贷记行本次清算前有余额预警，且本次清算后贷记行所有子账户余额加和大于余额预警值，

如果本次清算发生在子账户，则将子账户头寸调拨至主账户，用于余额预警解除；如果本次清算发生在主账户，则直接进行余额预警解除；

b) 如果本次清算发生在主账户，且借记行本次清算后导致余额首次低于余额预警值，清算系统将其他实例中子账户余额调拨至主账户，主账户余额仍旧达到余额告警值，则将余额预警告警通知报文给监视行；

7) 如果贷记行存在排队，本次清算发生在子账户，则将子账户可用头寸调拨至主账户用于排队解救，本次清算发生在主账户，则直接进行排队解救；

8) 清算系统发送清算回执报文到交易系统，处理状态为“清算”，转(11)；

9) 如果当前清算系统为子账户，则将清算报文路由至主账户实例中处理，主账户可用头寸仍旧不足时，将子账户中的可用头寸调拨至主账户，仍旧不足时，将该清算业务纳入清算排队；

10) 清算系统发送清算回执报文到交易，处理状态为“排队”；

11) 处理流程结束。

2.2.3. 模型设计

本算法模型设计遵循对参与者透明，维持对参与者外特性不变这一基本设计原则。

分账户方案，将清算账户分为一个主账户(原清算账户，所有清算账户的主账户在同一个实例中)和多个子账户，子账户数是可变的，根据商业银行可用头寸、业务量情况等特点可动态配置。

1) 账户设置和使用策略

各清算账户根据商业银行可用头寸、业务量情况等特点动态配置可拆分的子账户的数量，包括一个主账户，和任意数量的子账户(可为 0)。子账户的设置和修改由客户端触发，子账户的拆分个数可根据一段时间内(如 30 天)平均业务量峰值情况，综合考虑机器处理性能来调整应拆分的子账户个数。

子账户个数 = 平均业务量峰值/单台机器的处理性能。

对于流动性相关的业务，如排队等的设计，仅保存在主账户中。风险预警与监控可以仅监控主账户，当主账户可用头寸达到清算账户余额下限时，触发头寸调拨或可用头寸重新分配。

对于常见的涉及到双边的借贷记业务，通过某种算法(参见本章影子账户选取章节中说明)路由到相应的实例，选择本次要使用的借贷记方的账号。

对于涉及到多边的业务，如同城净额业务资金清算，此类业务量较小，可以路由到主账户中处理。同时，根据具体业务场景，可综合考虑跨实例交互处理方案。

2) 账户部署方案

如图 5 所示，所有行的主账户均部署在同一个实例(主实例)中，其他的子账户可根据负载均衡平均部署到其他实例中。实例 1 包含所有行的主账户，其他实例包含部分行的子账户。可统计一段时间内，账户间业务交互次数 $\sum_{t1}^{t2} k(A,B)$ ($t1$, $t2$ 时间段内，账户 A,B 行间交互次数)，根据统计结果设定阈值 K。将一段时间内行间业务交互次数高于阈值 K 的组成行号组，将其子账户组分配到同一个实例中。如银行 A、银行 B、银行 C、银行 D、银行 E 间发生业务频率较高，那么将五大行作为一个组合统一部署到同一个业务库中，如(A2、B2、C2、D2、E2)部署到业务实例 2 中。

3) 影子账户选取

考虑到分布式事务效率较低，同一笔业务尽量路由至同一个实例中处理。根据发生业务交互的行号组，确定可路由的业务实例集合{A 行-B 行，(业务实例 1，业务实例 2,..)}，当可路由的业务实例集合中包含超过一个业务实例时，按交易序号进行负载均衡(同时可考虑针对业务发生金额路由至不同的实例中)，路由至具体的业务实例中。

Figure 5. Account deployment
图 5. 账户部署方案

如图 6 所示，以银行 A 拆分成一个主账户 A1，两个子账户 A2、A3，银行 B 拆分成一个主账户 B1，一个子账户 B2，银行 C 拆分成一个主账户 C1，一个子账户 C2 为例来说明影子账户路由策略。如图 6 所示，业务实例 1 是主账户实例，包含所有行的主账户，为保证负载均衡，银行 A 子账户 A2、A3 分别分布在业务实例 2 和业务实例 3 上，银行 B 子账户 B2、银行 C 子账户 C2 分别分布在业务实例 2 和业务实例 3 上。当一笔业务涉及到超过一个行时，需要以行号组对作为路由规则。银行 A 银行 B 间的业务可以路由到业务实例 1 和业务实例 2 中 {银行 A-银行 B，(业务实例 1，业务实例 2)}，银行 A 银行 C 间的业务可以路由到业务实例 1，业务实例 3 中{银行 A-银行 C，(业务实例 1，业务实例 3)}，银行 B 和银行 C 间的业务可以路由到业务实例 1 中{银行 B-银行 C，(业务实例 1)}，因此路由组件根据行号对，首先确定可路由的业务实例集合，然后根据负载均衡策略路由到具体的业务实例。

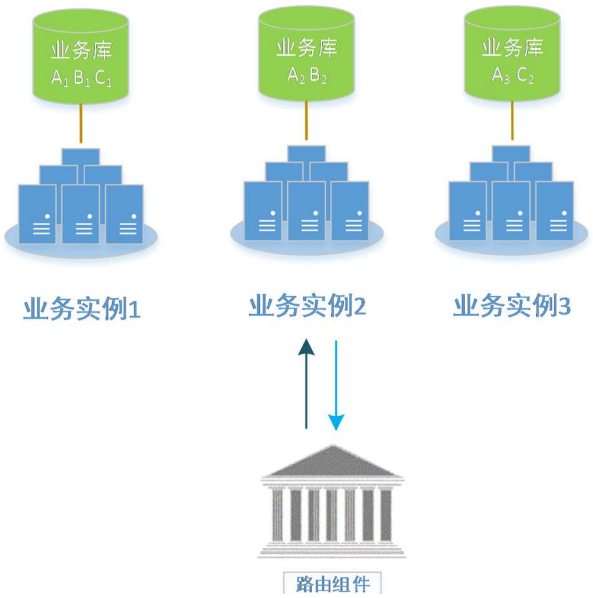

Figure 6. Routing strategy
图 6. 路由策略

4) 可用额度分配

对于分立出子账户的参与者，主账户和子账户共享可用额度。

a) 额度分配方案一

主账户分配较大比例的可用额度，子账户分配较小比例的可用额度。

当前可分配额度 = 所有账户当前可用额度；

主账户分配额度 = 当前可分配额度 × 主账户分配比例；

主账户分配比例可以根据一段时间内(如 30 天)发生的业务金额统计结果来确定，将发生的业务金额根据聚类算法[3] (聚类算法是把具有相同特征的数据聚集在一组，每一组都会有一个中心点，然后计算数据到每个中心点的距离，再将该数据分配到距离最小的那个中心点所属的组中)分为大金额类和小金额类。根据大金额类中心点 K1 和小金额类中心点 K2 的比例，计算主账户应该分配的可用额度比例。

主账户分配比例 = K1/(K1 + K2)；

子账户分配额度 = (1−主账户分配比例)×当前可分配额度/子账户数目。

该分配模式下，大金额业务路由至主账户，小业务金额路由至子账户，路由机制取决于业务发生的金额大小，需要修改内部报文，将需要路由的实例号写入报文头中。

b) 额度分配方案二

主账户看作一个特殊的子账户，分配的可用额度是子账户分配额度的 K 倍，K 可根据额度分配后头寸不足发生的概率进行相应的调整，当头寸不足发生概率较大时，将系数调减，降低头寸不足，报文重路由发生的概率。

当前可分配额度 = 所有账户当前可用额度；

主账户分配比例 = K/(子账户个数 + K)；

主账户分配额度 = 当前可分配额度 * 主账户分配比例；

子账户分配额度 = 当前可分配额度/(子账户个数 + K)；

由于业务金额的随机性，一段时间后，各账户之间的金额分布可能较初始分配时发生较大的变化，为避免各账户可用余额偏离初始分配比例过大，导致业务拒绝或小额排队业务，在固定间隔和特定事件(如业务路由至某个子账户，因可用额度不足需要路由至主账户处理)发生时重新进行可用额度分配。

3. 总结

借贷分离方案可以解决应用分布式改造，各实例分别负责部分账户的记账处理，可以在一定程度上提高业务连续性，某节点故障时，只影响部分行的业务处理，其他行可正常受理业务。借贷分离无法解决热点账户问题，且引入的分布式事务，对处理效率有一定的影响。对于交互度不够密切的行之间，借贷记账户位于不同的实例中，该场景下需借助借贷分离方案来解决。

影子账户方案可以解决应用分布式改造，各实例分别负责部分子账户的记账处理；可以解决热点账户问题，将热点账户业务拆分到不同的子账户中处理；可以较大程度提高业务连续性，某节点故障时，可以由其他子账户受理业务。由于账户被拆分成多个子账户，当某一子账户出现头寸不足时，需要进行头寸调拨。由于支付系统各账户可用头寸充足，该方案可以较大程度的避免分布式事务，提高处理效率。

支付系统包含的清算账户数量较少，针对支付系统可用头寸较充足的业务特点，支付系统账户系统分布式改造主要业务场景适于采用影子账户方案，针对某些特殊的业务场景，如交互度不密切的行之间业务处理，子账户头寸不足时头寸调拨(可以理解为内部的借贷分离操作)等可以借助借贷分离方案来实现。

参考文献

[1] 程立. 大规模 SOA 系统中的分布事务处理[EB/OL]. https://wenku.baidu.com/view/be946bec0975f46527d3e104.html, 2018-7-2.

[2] 李林峰. 分布式服务框架原理与实践[M]. 北京: 电子工业出版社, 2016: 286-289.

[3] Jiawei Han, Micheline Kamber. 数据挖掘概念与技术[M]. 北京: 机械工业出版社, 2007.

知网检索的两种方式：

1. 打开知网首页：http://cnki.net/，点击页面中“外文资源总库 CNKI SCHOLAR”，跳转至：http://scholar.cnki.net/new，搜索框内直接输入文章标题，即可查询；
   或点击“高级检索”，下拉列表框选择：[ISSN]，输入期刊 ISSN：2161-8801，即可查询。
2. 通过知网首页 http://cnki.net/顶部“旧版入口”进入知网旧版：http://www.cnki.net/old/，左侧选择“国际文献总库”进入，搜索框直接输入文章标题，即可查询。

投稿请点击：http://www.hanspub.org/Submission.aspx
期刊邮箱：csa@hanspub.org