5. **多源异构数据管理构件**

**5.1 构件功能简述**

为了结合区块链本身的特性，贯彻“以链治链，多链互通”的核心技术思想，实现面向大规模制造产业链全域异构数据协同可信管理需求，需要基于区块链的多源异构全域标识数据管理架构，包括层次化的区块链模型和高效的动态跨链可信查询方法。该架构应包含供应商链、制造商链、运输商链和经销商链等4个子链。每个子链的节点都有自己的链下数据库。所有子链也可以作为主链的链下数据库。主链由若干个节点来共同维护，从而形成层次链模型。层次型异构区块链架构，通过高并发集成实现全域互通，促进多源异构数据统一管理，并设计支持多种语义的可信查询方法。针对于联盟链由若干机构联合发起，由盟友共同来维护，它只针对特定某个群体的成员和有限的第三方开放的特点，解决联盟链跨链身份管理严格，交互关系复杂，链间信息相互隔离导致可信、高效、安全跨链的难以实现的关键问题，设计多源异构数据管理构件用于实现层次链间的跨链互通。



图5.1 多源异构数据管理模型

**5.2 构件架构设计**

**5.2.1 跨链模型**

跨链系统包括注册模块、申请审批模块、跨链请求模块、中继链模块、中继链监控模块、任务模块、消息模块等功能模块，如表5.1 所示。

注册模块、申请审批模块、跨链请求模块会通过中继链模块把数据写入中继链，完成基本信息的记录，这些信息用于后续指导跨链任务的顺利执行。当外部调用（业务链调用）开始后任务模块会将任务信息写入中继链，中继链产生数据后被中继链监控模块发现，进而推动任务执行下一个步骤，然后向中继链写入数据……该过程直到任务执行完毕或出现异常而终止，然后把结果发送到消息模块。最终用户通过订阅消息得到跨链交易执行的最终结果。

**表5.1 模块及功能**

|  |  |
| --- | --- |
| 模块 | 主要功能 |
| 注册模块 | 主要完成业务链信息的注册，这些信息包括链信息、链身份、想要跨链的业务链合约及方法等 |
| 申请审批模块 | 用于跨链交易模式申请及审批，模式经审批后，该类跨链交易才可以调用。交易中的任何一方可随时改变模式的状态，进而使得后续该类交易不可正常进行 |
| 跨链请求模块 | 跨链交易进行过程中，用于在业务链上执行链内交易 |
| 中继链模块 | 进行中继链上各种数据的读写和校验，表现为中继链上的跨链智能合约 |
| 中继链监控模块 | 监控中继链上相关的数据的变化，进而采取对应的操作 |
| 任务模块 | 用于串起跨链交易的各阶段并推进跨链交易按步骤执行 |
| 消息模块 | 用于传递跨链交易执行的结果 |

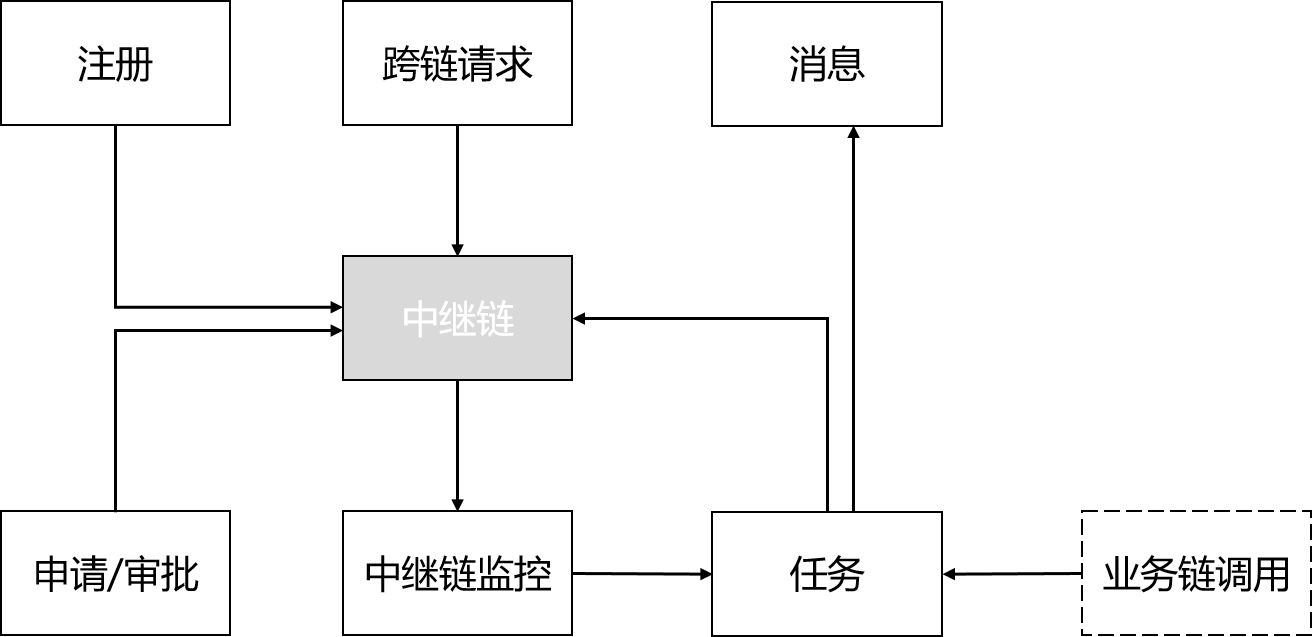


图5.2 核心模块

在系统实现的过程中，采用了一些已有的技术及新研组件作为基础，已有技术参见表5.2：

**表5.2 采用的已有技术**

|  |  |
| --- | --- |
| 名称或分类 | 本系统中的作用 |
| 缓存系统 | 交易摘要等数据都位于中继链上，存取效率相对较低，采用缓存系统用于提升访问效率 |
| 消息队列 | 完整的跨链交易是异步过程，交易发起方采用订阅方式获取最终结果，极大提高系统响应能力 |
| 加解密 | 密码学在区块链的应用，在本系统中主要作用是区块链访问、安全、传输安全、身份鉴别、防篡改、防抵赖 |
| 格式转换 | 用于异构数据之间的转换 |
| 自研组件 | 链访问SDK：结合密钥进行区块链访问；事件总线：用于访问解耦；链通用包：区块链通用组件。 |

新增组件参见表5.3：

**表5.3 新增自研组件**

|  |  |
| --- | --- |
| 名称或分类 | 本系统中的作用 |
| ID生成器 | 分布式递增ID的生成，在大部分情况下会使中继链数据查询更高效 |
| 流程监控 | 用于跨链交易的推进及监控 |
| 数据统计 | 用于数据统计、可视化展现，不依赖中继链 |
| 跨链授权 | 用于业务链方基于角色的跨链访问控制 |
| 事件总线 | 系统解耦，降低系统各部分的依赖 |
| 数据检索 | 独立于中继链的全文检索 |

**5.2.2 跨链交互**

#### **交互流程**

系统首先提供了注册的功能，各业务链将有跨链需要的链信息、合约及方法信息、链身份等通过注册功能登记到中继链；然后由需要跨链交易的发起方编排跨链交易，分解为单方或双方在各自业务链上有序执行交易的过程，并将结果作为申请登记到系统待对方审批，此时系统会通过交易约束检测系统检查该申请是否满足交易约束；对方复核后发现没有问题后通过申请。在注册用于执行交易的业务链合约和方法时，可进一步选择注册该方法的补偿方法，用于将来跨链交易调用发生异常时做补偿处理。

业务链跨链交互流程如图5.3 所示：



图5.3 跨链交互流程

假定要执行跨链交易A->B则主要流程如下：

1. A发起交易申请：可以由链A上的App通过系统接口发起申请；
2. 双向代理A在链A发起交易，并将交易输出及摘要入中继链：双向代理A结合链A的SDK在链A上完成交易，并结合中继链的SDK在中继链上执行任务；
3. 双向代理B发现交易，验证已完成部分的正确性，然后在链B发起交易，并将交易输出及摘要入中继链：双向代理B利用存储在中继链上的公钥校验双向代理A写入的数据，结合链B的SDK在链B上完成交易，并结合中继链的SDK在中继链上执行任务；
4. 双向代理A发现B完成了交易，验证B完成交易的正确性，并最终给出执行结果：双向代理A利用存储在中继链上的公钥校验双向代理B写入的数据；
5. A接受执行结果：A通过消息系统接收返回结果。

在上述流程中，系统是如何鉴别交易申请，并保证不同交易以及同类交易的不同实例之间结果不发生干扰的呢？实际上在发起交易申请时，系统首先会检查发起是否合规，例如是否是已通过批准的申请、是否是正确的发起方等，通过检查后系统会创建一个申请待执行的实例——本文称之为任务，该任务具有唯一的ID，贯穿整个交易执行过程。正是通过这唯一的ID保证了不同任务之间不会干扰。

#### **中继链监控**

中继链数据监控流程如下：

步骤：

1. 建立中继链数据监听机制；
2. 中继链产生交易数据Tk，推送到中继数据监听器；
3. 中继数据监听器对数据Tk进行分类，然后分发到事件总线；
4. 系统存在多种预先建立的事件，并基于事件监听机制建立了多种事件监听器，各事件监听器监听事件总线上的特定事件；
5. 事件监听器监听到需要处理的事件，以该事件作为输入，根据各自的逻辑调用不同的服务来处理输入，产生输出。



图5.4 中继链监控

以链注册为例：

1. 中继链将交易数据Tk推送到中继数据监听器；
2. 中继数据监听器对Tk进行分析识别，发现是链注册数据，然后组装成ChainEvent事件chainEvent1，发送到事件总线；
3. 系统预先针对ChainEvent注册了事件监听器ChainEvent Listener，当事件总线上出现该类事件后，ChainEvent Listener接收到chainEvent1；
4. ChainEvent Listener调用相关服务进行后续处理，如：更新统计数据、清理链相关缓存等等。

#### **交易感知及任务监控**

在执行跨链交易过程中，发起方何时发起交易是随机的，而且系统中交易涉及到的两个异构链也无法直接通信，因此我们必须找到一种方法或机制让交易另一方能够知道有需要它参与的交易在执行；另一方面，一个任务从开始到完成也会经历很多个过程，也需要一个监控系统来监视并推动任务的执行。本文采用了基于中继链区块数据监听机制来感知交易，并在得知交易发生后建立起任务监控系统来监视任务。



图5.5 交易感知及任务监控示意图

如图5.5，假设某个交易全部完成需要在中继链上存储，，…，，发起方发起跨链交易时，系统通过验证后会将初始信息写入，此时位于双向代理节点上的区块链监听模块监听到交易初始信息，任务监控系统开始工作，先根据之前已通过审批的申请信息在监控中模拟建立跨链交易在中继链中需要存储的一系列状态，，…，，然后再根据申请信息判断下一步是否由本代理节点推动任务向下一个状态进行，如果下一步是由本节点来推动，则调用相应的方法处理，并把结果写入下一个交易；区块链监听模块发现变化后，再传递到任务监控系统，监控系统与预先模拟建立的状态对比，查看是否一致……，以此类推，直到预先模拟的状态全部执行完毕。执行过程中发生异常时，任务监控系统会根据当前状态决定是否启动本方的补偿调用。

任务监控系统执行某个任务，直到成功或失败，会将结果发送到消息系统，最初调用方通过订阅消息得知最终结果，同时监控系统会在内存中把该任务监控清理掉。

#### **任务监控**

任务监控数据接收同样是通过中继链监控，所以前面数据获取部分不再赘述，此处只讲解后续监控处理流程，如下：

1. 中继链监控将产生的数据推送到中继链数据监听器，中继链数据监听器对数据进行分类并分发到事件总线；
2. TaskEvent Listener监听到任务事件；
3. TaskEvent Listener调用任务监控服务接口的更新监控方法；
4. 更新监控方法先查看当前监控对象，不存在则创建该对象。监控对象存有预先创建的状态列表的虚拟实例（FakeInstance），同时有当前状态的指针指向最新状态；
5. 更新监控方法以任务事件为输入更新当前监控对象的当前状态；
6. TaskEvent Listener调用DoNext方法推进交易进行，DoNext从当前监控对象得到下一个要经历的状态，然后进行相关操作，完成后将数据写入中继链；
7. 重复1) 直到监控对象的当前状态到达虚拟实例的结尾或出现异常。



图5.6 任务监控

以某个跨链交易为例，见图5.6：

1. 中继链写入初始状态Tinit，推送到中继链数据监听器，然后分发到事件总线，被TaskEvent Listener接收到；
2. TaskEvent Listener调用任务监控服务接口的更新监控方法Update；
3. Update查看当前监控对象，发现不存在则创建该对象，则生成虚拟实例的完整状态列表Tinit、Tfrom、Tfquery、Ttverify、Tto、Ttquery、Tfverify、Tend，更新CurrenStatus为Tinit；
4. TaskEvent Listener调用DoNext，从监控对象得到下一状态Tfrom，进而调用相关服务去业务链执行交易，完成后组装交易数据，状态为Tfrom，写入中继链；
5. 执行第1步；
6. 执行第2步，当前监控对象存在，则直接更新CurrenStatus为Tfrom；
7. 执行第3步，组装数据，状态为Tfquery，写入中继链；
8. 重复执行1~3，直到状态Tend，或出现异常；
9. 后续清理环境，如清除当前监控对象，或出现异常时调用补偿方法等。

**5.3 构件接口规范**

**5.3.1 应用链管理**

#### **注册链**

* 请求地址：/chain/register
* 请求方法：POST
* 请求参数：

输入参数：

* + activity：活动状态
  + chainDesc：链描述
  + chainName：链名称
  + chainType：链类型
  + description：描述
  + editable：编辑标志位
  + extension：拓展
  + id：合约ID
  + lifeCycleStatus：生命周期状态
  + identity：身份公钥
  + identities：身份公钥

#### **修改链**

* 请求地址：/chain/modify
* 请求方法：POST
* 请求参数：

输入参数：

* + activity：活动状态
  + chainDesc：链描述
  + chainName：链名称
  + chainType：链类型
  + description：描述
  + editable：编辑标志位
  + extension：拓展
  + id：合约ID
  + lifeCycleStatus：生命周期状态
  + identity：身份公钥
  + identities：身份公钥

#### **查询链**

* 请求地址：/chain/all
* 请求方法：GET

**5.3.2 跨链管理**

#### **申请单方跨链**

* 请求地址：/apply/one
* 请求方法：POST
* 请求参数：

输入参数：

* + fromContractId：节点名称
  + fromMethodId：组织ID

#### **申请双方跨链**

* 请求地址：/apply/two
* 请求方法：POST
* 请求参数：

输入参数：

* + fromContractId：节点名称
  + fromMethodId：组织ID

#### **跨链申请审批**

* 请求地址：/apply/approved/approve
* 请求方法：POST
* 请求参数：

输入参数：

* + fromContractId：节点名称
  + fromMethodId：组织ID

#### **跨链申请驳回**

* 请求地址：/apply/approved/reject
* 请求方法：POST
* 请求参数：

输入参数：

* + fromContractId：节点名称
  + fromMethodId：组织ID

#### **跨链申请复核**

* 请求地址：/apply/approved/check
* 请求方法：POST
* 请求参数：

输入参数：

* + fromContractId：节点名称
  + fromMethodId：组织ID

#### **跨链申请列表**

* 请求地址：/apply/list
* 请求方法：POST
* 请求参数：

输入参数：

* + admin：admin身份识别
  + bookmark：辅助查询参数
  + fetchedCount：辅助查询参数
  + fromChainId：源链
  + identity：ip地址
  + identity：实体
  + pageNum：页数
  + pageSize：单页数据量
  + size：总数据量
  + toChainId：目标链

**5.3.3 跨链任务**

#### **注册合约**

* 请求地址：/method/register/one
* 请求方法：POST
* 请求参数：

输入参数：

* + chain：目标链参数
  + chainId：链ID
  + contract：合约名
  + creator：申请者
  + description：描述
  + editable：编辑标志位
  + extension：拓展
  + id：合约ID
  + lifeCycleStatus：生命周期状态
  + methodIds：方法ID
  + methods：方法
  + name：合约名

#### **删除合约**

* 请求地址：/contract/delete
* 请求方法：POST
* 请求参数：

输入参数：

* + chain：目标链参数
  + chainId：链ID
  + contract：合约名
  + creator：申请者
  + description：描述
  + editable：编辑标志位
  + extension：拓展
  + id：合约ID
  + lifeCycleStatus：生命周期状态
  + methodIds：方法ID
  + methods：方法
  + name：合约名

#### **修改合约**

* 请求地址：/contract/modify
* 请求方法：POST
* 请求参数：

输入参数：

* + chain：目标链参数
  + chainId：链ID
  + contract：合约名
  + creator：申请者
  + description：描述
  + editable：编辑标志位
  + extension：拓展
  + id：合约ID
  + lifeCycleStatus：生命周期状态
  + methodIds：方法ID
  + methods：方法
  + name：合约名

#### **查询合约**

* 请求地址：/method/register/one
* 请求方法：POST
* 请求参数：

输入参数：

* + bookmark：辅助查询参数
  + chainId：链ID
  + contractName：合约名
  + fetchedCount：辅助查询参数
  + pageNum：分页数
  + pageSize：分页尺寸
  + size：数据总量

#### **注册合约方法**

* 请求地址：/method/register/one
* 请求方法：POST
* 请求参数：

输入参数：

* + args：参数
  + compensate：跨链补偿
  + compensateRef：补偿引用
  + contractId：合约ID
  + creator：创建者
  + editable：编辑标志位
  + funcName：函数名
  + id：方法ID
  + lifeCycleStatus：生命周期
  + methodDesc：方法说明
  + methodName：方法名
  + methodType：方法类型
  + returnType：返回类型

#### **修改合约方法**

* 请求地址：/method/modify/one
* 请求方法：POST
* 请求参数：

输入参数：

* + args：参数
  + compensate：跨链补偿
  + compensateRef：补偿引用
  + contractId：合约ID
  + creator：创建者
  + editable：编辑标志位
  + funcName：函数名
  + id：方法ID
  + lifeCycleStatus：生命周期
  + methodDesc：方法说明
  + methodName：方法名
  + methodType：方法类型
  + returnType：返回类型

#### **删除合约方法**

* 请求地址：/method/delete
* 请求方法：POST
* 请求参数：

输入参数：

* + args：参数
  + compensate：跨链补偿
  + compensateRef：补偿引用
  + contractId：合约ID
  + creator：创建者
  + editable：编辑标志位
  + funcName：函数名
  + id：方法ID
  + lifeCycleStatus：生命周期
  + methodDesc：方法说明
  + methodName：方法名
  + methodType：方法类型
  + returnType：返回类型

#### **查询合约方法**

* 请求地址：/method/list
* 请求方法：POST
* 请求参数：

输入参数：

* + id：合约ID

#### **调用跨链**

* 请求地址：/cross/call
* 请求方法：POST
* 请求参数：

输入参数：

* + id：合约ID

**5.3.4 授权管理**

#### **添加授权信息**

* 请求地址：/authorize/add
* 请求方法：POST
* 请求参数：

输入参数：

* + applicationId：申请ID
  + applicationName：申请名
  + authorizeId：授权ID
  + authorizeName：授权名
  + createTime：创建时间
  + params：辅助参数
  + roleId：角色ID
  + sysRole：系统角色
  + updateTime：更新时间

#### **编辑授权信息**

* 请求地址：/authorize/edit
* 请求方法：POST
* 请求参数：

输入参数：

* + applicationId：申请ID
  + applicationName：申请名
  + authorizeId：授权ID
  + authorizeName：授权名
  + createTime：创建时间
  + params：辅助参数
  + roleId：角色ID
  + sysRole：系统角色
  + updateTime：更新时间

{

"applicationId": "string",

"applicationName": "string",

"authorizeId": 0,

"authorizeName": "string",

"createTime": "2022-09-28T00:58:25.978Z",

"delFlag": "string",

"params": {},

"roleId": 0,

"sysRole": {

"createBy": "string",

"createTime": "2022-09-28T00:58:25.978Z",

"dataScope": "string",

"delFlag": "string",

"deptCheckStrictly": true,

"deptIds": [

0

],

"flag": true,

"menuCheckStrictly": true,

"menuIds": [

0

],

"params": {},

"remark": "string",

"roleId": 0,

"roleKey": "string",

"roleName": "string",

"roleSort": "string",

"searchValue": "string",

"status": "string",

"updateBy": "string",

"updateTime": "2022-09-28T00:58:25.978Z"

},

"updateTime": "2022-09-28T00:58:25.978Z"

}

* 请求示例：

curl -X POST "http://10.9.27.56:12000/authorize/add" -H "accept: \*/\*" -H "Content-Type: application/json" -d "{\"applicationId\":\"string\",\"applicationName\":\"string\",\"authorizeId\":0,\"authorizeName\":\"string\",\"createTime\":\"2022-09-28T01:24:00.035Z\",\"delFlag\":\"string\",\"params\":{},\"roleId\":0,\"sysRole\":{\"createBy\":\"string\",\"createTime\":\"2022-09-28T01:24:00.035Z\",\"dataScope\":\"string\",\"delFlag\":\"string\",\"deptCheckStrictly\":true,\"deptIds\":[0],\"flag\":true,\"menuCheckStrictly\":true,\"menuIds\":[0],\"params\":{},\"remark\":\"string\",\"roleId\":0,\"roleKey\":\"string\",\"roleName\":\"string\",\"roleSort\":\"string\",\"searchValue\":\"string\",\"status\":\"string\",\"updateBy\":\"string\",\"updateTime\":\"2022-09-28T01:24:00.035Z\"},\"updateTime\":\"2022-09-28T01:24:00.035Z\"}"

* 返回示例：

{

"msg":"操作成功",

"code":200

}

#### **删除授权信息**

* 请求地址：/authorize/remove/{authorizeIds}
* 请求方法：DELETE
* 请求参数：

输入参数：

* + authorizeId：授权ID

#### **查询授权信息**

* 请求地址：/authorize/list
* 请求方法：DELETE
* 请求参数：

输入参数：

* + authorizeId：授权ID