# 文档说明

此文档介绍了Lanchain平台在各场景下的扩容方案.

# 应用场景与扩容方案

## 2.1 竞拍场景扩容方案

### 1. 场景介绍

传统竞拍方式为拍卖者首先向拍卖行提供拍卖物,拍卖行会对拍卖物进行查验.对于符合上拍条件的拍卖物,拍卖行会在特定时间举行现场或者网络上的拍卖会,在拍卖会上,竞拍者通过在规定时间内提交具有竞争力的价格的方式来参与竞价,最终在拍卖结束的时候最高价格的竞价者获得拍卖品.

区块链上的竞拍流程与此类似.首先,拍卖者向区块链平台提交竞拍物品信息及发布竞拍信息.在指定的拍卖时间范围内,认证用户通过提交自己的竞拍价格的方式参与竞拍.用户能够通过平台接口周期性获得当前最高竞拍价格.最终,在拍卖时间结束时最高价格的用户获得拍卖品.

在区块链平台上进行竞拍具有防欺诈,过程透明可查等特点.

此竞拍场景扩容方案只关注竞拍过程的性能及吞吐率的提高,不关注存证(例如:竞拍品信息验证及存证)及身份认证过程(例如:只有授权用户才能发布竞拍信息及参与竞拍).

### 方案介绍

此扩容方案采用单一根链与多个子链的方式进行扩容,目的是提高竞拍场景中大量竞拍行为同时发生时平台的交易吞吐率,提高平台整体的交易tps.

方案中竞拍行为在各子链上发生, keeper服务周期性的扫描所有子链,从各子链获得一定时间范围内的最高竞价信息并更新到根链上.最终通过调用根链上的合约接口获得最终竞价信息.

方案特点如下:

* 联盟内共识方式管理子链,保证子链安全性及可信性.
* 通过随时增加子链的数量来达到动态扩充平台容量(吞吐率,整体tps等)的目的.
* keeper服务程序采用中心化或分布式方式以适用于不同安全环境.
* 主链的最高竞拍记录包含子链id,块hash及交易hash及keeper标识,方便数据快速查找及验证.

竞拍场景扩容方案示意图如下:

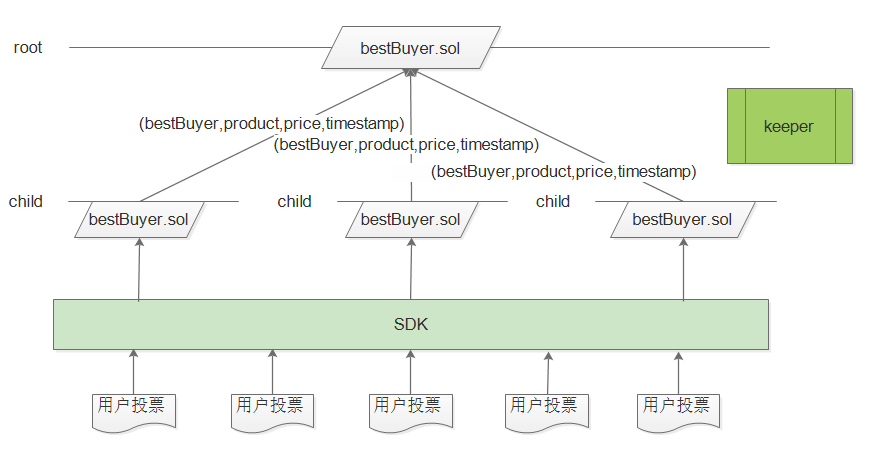


图1. 竞拍场景扩容方案示意图

方案中各部分说明如下:

* root: 根链,可以为wanchain等公链,也可以为指定联盟链.
* child: 子链.
* Keeper: 服务程序,负责子链与根链之间的数据更新,具体参见下文.
* bestBuyer.sol: 竞拍示例合约,同时部署在根链及所有子链上.
* SDK: 提供给用户的平台API. SDK需要提供在各子链间进行负载均衡的能力,具体参见下文.
* 用户投票: 用户通过调用平台SDK进行竞拍信息提交及参与竞拍,具体参见下文.

#### 子链管理

此方案中子链的创建及销毁需要在联盟内部通过一定共识算法达成共识.在达成共识之后联盟的各节点才能创建或者删除子链.

此方法能够维持子链在联盟中的可信性及稳定性.

子链共识结果存在根链的锚定合约anchor.sol中,此合约不止存储子链创建时的共识证明,还能够存储各子链的其他锚定信息(此方案中不需要此功能).

子链管理示意图如下:

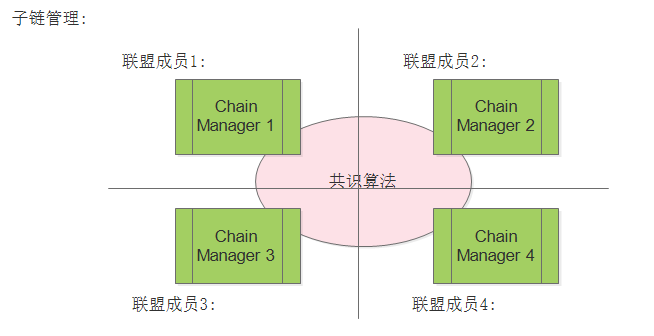


图2. 子链管理示意图

Anchor.sol合约主要成员说明如下:

*childChain:*

*String id: 子链id.*

*Int status: 子链当前状态: 运行,销毁.*

*Mapping History: history为一个映射.*

*String Action: 子链操作动作.*

*Mapping Signers: 各方签名,只有联盟成员的签名达到一定要求,action才会被执行.*

*Int timestamp: 时间戳.*

#### 竞拍交易部署

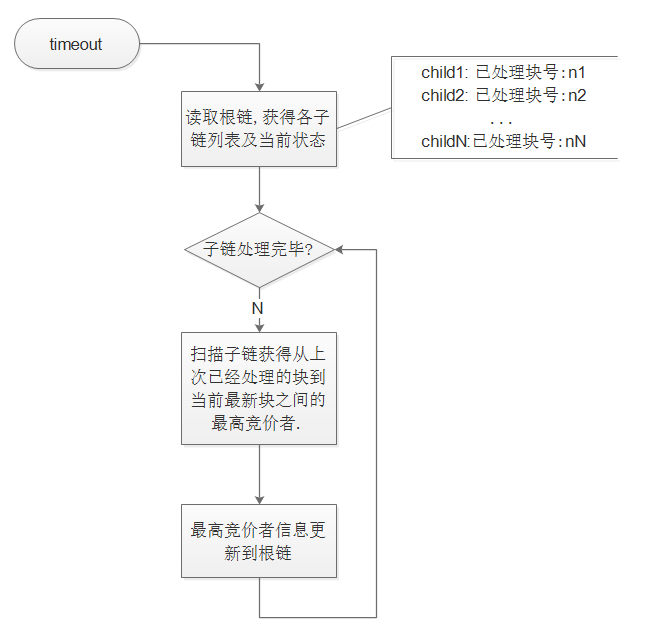
* 用户通过调用平台SDK完成竞拍交易的部署.
* SDK内部负责在所有当前正在运行的子链上部署竞拍交易并返回部署结果给用户.
* 用户对竞拍交易部署的部署过程及子链架构无感知.

#### **竞拍过程**

* 用户通过调用平台SDK参与竞拍.
* SDK根据各子链的负载进行负载均衡,把用户的竞拍请求均衡分配到不同子链上.
* 用户调用SDK接口监听竞拍价格更新通知.
* 用户对子链架构无感知.

#### **keeper服务相关**

* keeper服务运行流程如下:



图三. keeper处理流程.

* keeper周期性向主链提交的信息主要结构如下:

*bestBuyer {*

*buyerHash: 最高竞价者的唯一标识.*

*productHash: 竞品唯一标识.*

*Price: 竞拍价格.*

*timeStamp: 时间戳.*

*txHash: 最高竞价所在的交易hash.*

*blockHash: 最高竞价所在的块hash.*

*chainId: 最高竞价所在的子链ID.*

*signedKeepers: keeper签名(分布式keeper需要多方签名).*

*}*

* keeper中心化与分布式说明

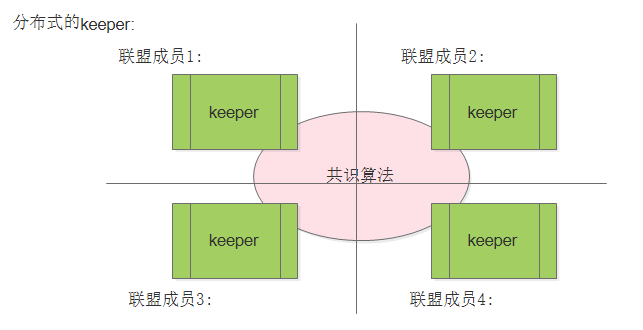
keeper作为一个链外服务程序可以采用中心化方式或者分布式方式部署,下面分别说明两种方式的优势及使用场景.

keeper中心化部署方式:

* 效率高.
* 部署及实现简单.
* 不能防止单keeper作弊(单点作弊).
* 虽然不能防止单点作弊,但是能够通过链上数据验证keeper是否作弊.

keeper分布式部署方式:

keeper分布式部署结构图如下:



图四. 分布式keeper结构图

分布式keeper特点如下:

* 多个keeper部署在联盟中不同成员中,保证分布性.
* 各keeper独立从各child链获取bestBuyer.
* keeper间通过共识协议对准备提交到主链的信息进行确认.
* 由提交节点负责提交bestkeeper到root chain上.
* 分布式方式能够防止单keeper作弊.

### 开放问题

## 2.2 身份验证场景方案

1. 场景介绍

2. <方案一>

2.1 <方案具体说明>

2.2 <方案优势及特点分析>

3. <方案二>

3.1 <方案具体说明>

3.2 <方案优势及特点分析>

4. <方案对比说明及综合分析>

## 2.3 存证场景扩容方案

1. 场景介绍

2. <方案一>

2.1 <方案具体说明>

2.2 <方案优势及特点分析>

3. <方案二>

3.1 <方案具体说明>

3.2 <方案优势及特点分析>

4. <方案对比说明及综合分析>

## 2.4 货币支付场景扩容方案

1. 场景介绍

2. <方案一>

2.1 <方案具体说明>

2.2 <方案优势及特点分析>

3. <方案二>

3.1 <方案具体说明>

3.2 <方案优势及特点分析>

4. <方案对比说明及综合分析>

## 2.5 供应链金融场景扩容方案

1. 场景介绍

2. <方案一>

2.1 <方案具体说明>

2.2 <方案优势及特点分析>

3. <方案二>

3.1 <方案具体说明>

3.2 <方案优势及特点分析>

4. <方案对比说明及综合分析>

# **专项说明**

### 3.1 链锚定方式

### 3.2 链外数据访问

# **引用参考**