StreamNet工程说明文档

文件编号：80303

版本编号：v.1.0.9

2019年02月27日

目录

[版本记录 1](#_Toc2263892)

[图片目录 1](#_Toc2263893)

[表格目录 2](#_Toc2263894)

[工程概述 2](#_Toc2263895)

[1 参数配置相关 3](#_Toc2263896)

[2 服务启动相关 4](#_Toc2263897)

[2.1 DAG服务的启动 5](#_Toc2263898)

[2.2 APP服务的启动 5](#_Toc2263899)

[2.3 Sync服务的启动 6](#_Toc2263900)

[3 HTTP服务接口相关 6](#_Toc2263901)

[4 SDK接口相关 7](#_Toc2263902)

[4.1 Python SDK 7](#_Toc2263903)

[4.2 Go SDK 7](#_Toc2263904)

版本记录

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **作者** | **修订说明** |
| v.1.0.0 | 2019-02-27 | 尹朝明 | 增加StreamNet工程概述和参数配置 |
| v.1.0.1 | 2019-02-28 | 尹朝明 | 增加服务启动/http服务接口 |

图片目录

图 1 StreamNet的分支示意。 3

图 2 下载并编译StreamNet代码。 5

图 3 部署的各个服务之间的关系示意。 5

图 4 启动DAG服务的命令示意。 5

图 5 启动app 服务的命令示意。 6

图 6 APP服务的配置文件示意。 6

图 7 sync服务启动命令示意。 6

图 8 sync服务对应的配置文件示意。 6

图 9 put\_file接口示意。 7

图 10 put\_cache接口示意。 7

表格目录

表 1 StreamNet DAG配置说明 3

表 2 StreamNet APP配置说明 4

表 3 StreamNet Sync配置说明 4

工程概述

为了支持基于POW的区块链系统的高吞吐量，一系列方法被提出，包含侧链，分片，混合链，DAG等多种方案。我们设计了TRIAS StreamNet，它是基于现有成熟DAG系统的全新设计，其针对现有系统容易出现双花和重放攻击、交易确认速度慢、观察者的引入导致中心化假设等问题，以图计算中的流式图计算为基础，利用到了Katz中心度的计算来获取DAG中的一条中心链，在这个中心链中的每一个块都拥有最大的Katz得分（不是GHOST法则）。围绕中心链，通过Conflux算法可以去中心化的获得一个总的全局序的链。当新的块加入的时候，它会选择两个前置tip块来进行批准，第一个为”父”tip块，第二个是使用蒙特卡洛随机游走得到的随机tip块。StreamNet支持配置随机游走的额外检查来避免双花和算力攻击。

我们的程序提供两个服务DAG服务和APP服务。StreamNet DAG 服务原则上不直接提供应用服务，其已有的接口与IOTA保持一致。对外的服务体现在APP服务上。工程主要包含参数配置、服务启动、HTTP服务接口、SDK接口等几个部分。其中：

* 参数配置

StreamNet的DAG服务的配置例子在iri/scripts/examples里有包含如何起单台或两台StreamNet。

StreamNet的APP服务的配置例子在iri/scripts/examples里有包含如何起单台或者两台APP服务。

* 服务启动

该部分主要是完成StreamNet相关各服务的启动，包括启动StreamNet的DAG服务、APP服务 、Sync进程（注意sync进程不是对外服务，它是一个专门轮询DAG服务中关于TRIAS转账/合约信息并负责同步给TM的进程）。

* HTTP服务接口

该部分主要是APP服务对外暴露的HTTP RESTFul API接口的说明 。其中APP服务一共提供三个应用：缓存TRIAS的TEE信息，缓存TRIAS的交易信息，StreamNet自己发行的NetCoin的交易请求。

* SDK接口

该部分主要是提供SDK接口，方便在代码中直接对APP服务中的功能进行调用 。

StreamNet工程在coding上的URL：<https://github.com/wunder360/iri>。StreamNet工程的开发分支：dev。StreamNet工程的最新发布分支：v0.1-streamnet。同步trias-lab官方代码库的分支：trias-lab/iri/dev。本文档只支持发布分支v0.1-streamnet中的内容。

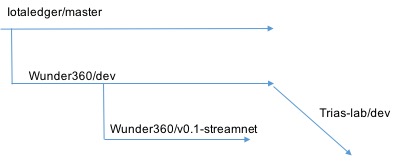


图 1 StreamNet的分支示意。

参数配置相关

StreamNet的参数配置包含了DAG服务相关的配置参数和APP服务相关的配置参数。DAG相关的参数可以通过命令行输入，也可以通过conf文件来配置。APP服务相关的参数需要通过conf文件来配置。Sync进程的相关参数需要通过conf文件来配置。

表 1 StreamNet DAG配置说明

|  |  |
| --- | --- |
| **参数名称** | **参数说明** |
| --testnet | 是否为测试网络，在公网上线前默认为这个选项 |
| --mwm | POW的难度，参数为int型数字，默认为14 |
| --walk-validator | 随机游走的检查方法，默认为”NULL”，表示不做检查 |
| --ledger-validator | 账本的检查方法，在v0.1-streamnet中默认为”NULL”，表示不做检查，因为这个功能正在完善中。 |
| -p  --udp-receiver-port  --tcp-receiver-port | HTTP的监听端口，负责RESTFUL api的调用接收  udp 端口，负责DAG之间的UDP通信  tcp 端口，负责DAG之间的TCP通信 |
| --remote | 是否接受远端服务调用请求 |
| --enable-streaming-graph | 是否启用流式图计算（能够起到加速的作用） |
| --entrypoint-selector-algorithm | 随机游走的初始块选择算法，默认为”DEFAULT”，这个需要观察者的引入，当选择”KATZ”的时候则没有此限制。 |
| --tip-sel-algo  --ipfs-txns | 随机游走的tip选择算法，MCMC为两个tip都是随机游走的，而CONFLUX则一个为parent，一个为随机游走。  是否将传入的信息存在IPFS中，这个主要针对信息过大的情况，需要IPFS的存储空间支持。 |

表 2 StreamNet APP配置说明

|  |  |
| --- | --- |
| **参数名称** | **参数说明** |
| addr | 连接的DAG的地址。 |
| seed | 发送消息时需要初始化的种子。 |
| enableIpfs | 是否将信息存入ipfs中，这个适用于信息量比较大的情况。 |
| enableBatching  enableCompression | 是否做批量化处理，这个适用于交易数据，做批量化处理可以增大吞吐。 |
| listenPort  listenAddress | HTTP的监听端口  HTTP的监听地址 |
|  |  |

表 3 StreamNet Sync配置说明

|  |  |
| --- | --- |
| **参数名称** | **参数说明** |
| addr | 连接的TM的地址。 |
|  |  |
|  |  |

服务启动相关

为了启动服务，首先需要将代码下载到本地，运行代码可以有三种方式，第一种方式是直接从发布分支（v0.1-streamnet）上直接编译运行，第二种方式是进入到一个稳定版本的发布（可以是在dev分支上，也可以是在发布分支上），下载对应的源码或者jar包，第三种方式是checkout到一个稳定的发布的tag上编译运行。这个文档中我们主要遵循第一种方式，因为StreamNet使用严格的Travis CI 进行持续集成控制，v0.1-streamnet上的功能基本上是稳定的。下载并编译代码的命令如图 2所示。本章节主要涉及到部署两个服务和一个进程，它们之间的关系如图 3所示。

1. git clone https://github.com/wunder3605/iri.git
2. cd iri
3. git checkout –b v0.1-streamnet origin/v0.1-streamnet
4. mvn clean
5. mvn package

图 2 下载并编译StreamNet代码。

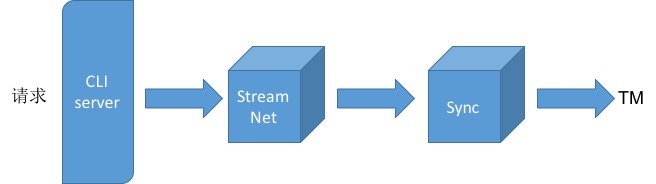


图 3 部署的各个服务之间的关系示意。

DAG服务的启动

当编译完成以后，DAG 服务只需要启动一个jar的server便可。命令如图 4所示。在这里，一个支持流式Katz图计算的，使用Conflux算法的，所有存储都在DAG内的StreamNet内核被启动起来。

1. cd target
2. java -jar iri-1.5.5.jar --testnet \
3. --mwm 1 \
4. --walk-validator "NULL" \
5. --ledger-validator "NULL" \
6. -p 14700 \
7. --max-peers 40 \
8. --remote \
9. --enable-streaming-graph \
10. --entrypoint-selector-algorithm "KATZ" \
11. --tip-sel-algo "CONFLUX" \
12. --ipfs-txns **false** \
13. &>  streamnet.log &

图 4 启动DAG服务的命令示意。

APP服务的启动

在这里，假设用户已经对如何使用python pip安装所需要的库有所了解。并回到了根目录中。使用图 5的命令能将APP服务启动。而此服务对应的配置文件在图 6中，这个服务会连到我们之前启动的DAG服务上，同时不支持IPFS，批量交易存储和压缩。它在本地的5000端口进行监听。

1. cd scripts/iota\_api/
2. gunicorn -w 4 app:app &> ../examples/cli.log  &

图 5 启动app 服务的命令示意。

1. [iota]
2. addr = http://localhost:14700
3. seed = EBZYNR9YVFIOAZUPQOLRZXPPPIKRCJ9EJKVCXMYVLMNOCCOPYPJKCWUZNLJZZZZWTMVQUXZFYLVLZXJ9Q
4. enableIpfs = False
5. enableBatching = False
6. enableCompression = False
7. listenPort = 5000
8. listenAddress = 0.0.0.0

图 6 APP服务的配置文件示意。

Sync服务的启动

启动sync服务的命令如图 7所示，其对应的配置文件如图 8所示。表示将会连接到本地的监听在26657端口上的TM服务。

1. cd iota\_api
2. python chronic\_txn\_sync.py

图 7 sync服务启动命令示意。

1. [tendermint]
2. addr = http://localhost:26657

图 8 sync服务对应的配置文件示意。

HTTP服务接口相关

原则上仅有APP服务对外提供Restful API的服务。目前提供服务的接口（endpoint）有两个，分别为put\_file图 9和put\_cache图 10。put\_file 支持上传单条json格式的数据，而put\_cache则支持将上传的json格式在本地缓存，在缓存达到一定的条数以后上传到DAG服务中。

1. curl -X POST \
2. http://address/put\_file \
3. -H 'Content-Type: application/json' \
4. -H 'cache-control: no-cache' \
5. -d '<json格式数据>'

图 9 put\_file接口示意。

1. curl -X POST \
2. http://address/put\_cache \
3. -H 'Content-Type: application/json' \
4. -H 'cache-control: no-cache' \
5. -d '<json格式数据>'

图 10 put\_cache接口示意。

SDK接口相关

Python SDK

在v0.1-streamnet版本中不支持python SDK。

Go SDK