【SKY-DP】FileCoin交易图谱分析与相关交易地址提取

1. 介绍

用户交易流程可以看成一个图结构（有向有环图）。

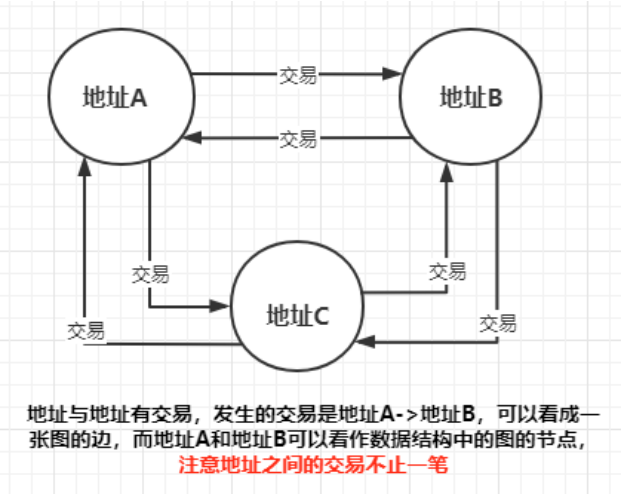
交易过程为图的边（属性: from 地址， to地址，总共金额，单比最大金额，单比最小金额，

总共gas 消耗，交易数量，最后交易时间，开始交易时间)，

地址为图的节点(属性：地址，获得的总共钱，获得的最大金额，支付的总共金额，

支付的最大金额，余额，总共gas消耗 ，入度（总共收入笔数），出度（总共支出笔数），转入地址数，转出地址数，转入开始时间， 转出开始时间，转入最近时间，

转出最近时间)



(图一)

2. 交易图结构分析的问题

问题:filecoin库中的全数据表，一行表示from->to的一条交易记录，所以一行数据可以看成图结构一条边，全表大约有10亿条数据，进行地址分析比较复杂(没有地址汇总信息)，且相同地址重复交易次数多，难以直接进行图结构分析。

3. 交易图结构优化

1. 优化边数据（优化all_msg表）

方法：由于地址与地址之间会有大量的重复交易，所以可以对地址间的重复数据进行聚合，但要统计 **from 地址， to地址，总共金额，单比最大金额，单比最小金额，总共gas 消耗，交易数量，最后交易时间，开始交易时间** 这些数据，就能全面的表示两地址之间的累计交易情况，从而使多条相同流向的边变成了一条

实现步骤：进行 **from-to**的聚合

SQL

```sql
create table EDG_table
as
SELECT `from`,`to`, sum(value) as total_value, max(value) as max_value,
min(value) as min_value, sum(total_cost) as total_gas_consume, count(*) as
count, max(block_height_at) as last_time, min(block_height_at) as start_time
from all_msg
WHERE exit_code = 0    //过滤交易失败的数据
GROUP BY `from`, `to`
having total_value > 0;  //过滤空交易的边
```

结果：10亿条数据----> 300万条数据，数据量减少300倍

2. 得出地址节点的出度相关数据(转出虚拟币的相关信息)

方法: 对from地址是转出地址，主要进行转出的一些地址，由于上面进行了对相同的边的聚合，并记录了相同边的条数以及相关数据，所以这一步对from进行聚合时，可以得到节点的

总共的支付价值，最大的支付价值，总共gas消费量，出度(进行支付的次数)，支付地址数，最近一笔支付时间，第一笔支付时间

实现步骤：对from进行聚合

SQL

```sql
CREATE TABLE VertexOutDegree
as
SELECT `from`, sum(total_value) as total_pay_value, max(max_value)
pay_max_value ,sum(total_gas_consume) pay_total_gas_consume,sum(count) as
outDegree, count(*) as pay_address_count, max(last_time)
pay_address_last_time, min(start_time) pay_address_start_time
from edg_table
GROUP BY `from`;
```

结果：得到节点的支付总数据

3. 得出地址节点的入度相关数据(转入虚拟币的相关信息)

方法: to地址是转入地址，主要进行转入的一些地址，由于上面进行了对相同的边的聚合，并记录了相同边的条数以及相关数据，所以这一步对to进行聚合时，可以得到节点的

总共的获得价值，最大的获得价值，入度(进行获取的次数)，获取地址数，最近一笔获取时间，第一笔获取时间

实现步骤：对to进行聚合

SQL

```sql
CREATE TABLE VertexInDegree
as
SELECT `to`, sum(total_value) as total_get_value, max(max_value) get_max_value
,sum(count) as inDegree, count(*) as get_address_count, max(last_time)
get_address_last_time, min(start_time) get_address_start_time
from edg_table
GROUP BY `to`;
```

4. 对地址的转入转出信息进行汇总，得到节点的总交易信息

方法: 有b.c步骤后可以得到两张关于节点的转入转出地址表，而from和to相同的数据行就是同一个节点的交易（转入转出）数据。可以得到节点的

地址，获得的总共价值，获得的最大价值，支付的总共价值，支付的最大价值，余额，

总共gas消耗 ，入度（总共收入笔数)，出度（总共支出笔数)，转入地址数,转出地址数,

转入开始时间,转出开始时间, 转入最近时间, 转出最近时间

实现步骤：对to,from进行join连接进行聚合

SQL

```sql
CREATE TABLE Vertex
as
SELECT
     coalesce(`from`, `to`) as filecoin_address,
     coalesce(total_get_value, 0) as get_total_value,
     coalesce(get_max_value, 0) as get_max_value,
     coalesce(total_pay_value, 0) as pay_total_value,
     coalesce(pay_max_value, 0) as pay_max_value,
     (coalesce(total_get_value, 0) - coalesce(total_pay_value, 0)) as extra_money,
     coalesce(pay_total_gas_consume, 0) as pay_total_gas_consume,
     coalesce(indegree, 0) as indegree,
     coalesce(outdegree, 0) as outdegree,
     coalesce(get_address_count, 0) as get_address_count,
     coalesce(pay_address_count, 0) as pay_address_count,
     coalesce(get_address_start_time, null) as get_start_time,
     coalesce(pay_address_start_time, null) as pay_start_time,
     coalesce(get_address_last_time, null) as get_last_time,
     coalesce(pay_address_last_time, null) as pay_last_time
FROM vertexindegree FULL JOIN vertexoutdegree ON `to` = `from`;
```

结果：对历史数据进行分析，可以得到100万个有价值的地址，对地址的余额与官网进行验证比较，发现精确度比官网高，更准确(除分裂节点)

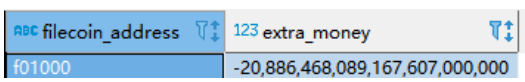
5. 结论

改进情况：对图进行处理过后，将10亿条数据进行了清洗，变成了300万条不同的聚合边数据，100万条不同的地址汇总数据。且进行优化过后，可以很容易的找到一些交易量大，交易数目多，余额多的节点，并作标记（方便找到交易所和洗钱节点），且大大加块了前后端对数据的查询的速度。

注意：对于节点余额数据有存在负值的情况，有1万多个节点，

1.可能地址为奖励地址

2.节点分裂，一个地址有多个地址字段表示，多个地址ID表示一个地址(已找到相关情况)



f01001	-24,371,408,307,552,170,000,000
f010013	-50,000,000,000,000,000,000
f010017	-7,786,667,211,339,651,100
f010019	-100,000,000,000,000,000,000
f010040	-2,048
f0100877	-4,116,000,000,000,000,000,000
f0100882	-1,836,000,000,000,000,000,000
f0100884	-1,575,000,000,000,000,000,000

4. 异常节点分析(洗钱节点)

1. 前提

由上述过程，我们得到了交易图谱的聚合边信息以及地址节点信息,且数据量不大

2. 判断洗钱规则

判断filecoin中是否存在洗钱，前提是确定洗钱的主体是地址,过程是通过图的边转到其他地址。洗钱是通过地址之间转虚拟币实现的，所以分析的主要方法是**异常地址的交易过程**

给地址判定为异常洗钱节点,边为异常交易过程需确定如下规则：

1.有钱即有罪：

1.余额大，交易量多的地址为嫌疑犯。交易量小，余额小的最多只存在中间转移的节点功能，不存在洗钱的性质，需要过滤

2.聚合边代表的是两人的总共交易过程，总共交易量小的边不构洗钱交易

2.犯罪必有同犯，单人犯不了罪：

1.只有余额大，交换量大的地址转向另外一个余额大交换量大的地址才有可能是洗钱，所以洗钱过程会有一个大金额地址转到另一个大金额地址

2.通过一个或者许多个中间边连接洗钱结点

3. 洗钱的判断机制

根据上述规则可以得到：洗钱的过程就是余额大，交易量多的地址节点把虚拟币转到另一个 或者几十个地址节点上

判断机制：只需要对金额量大的地址进行图连接计算，如果地址之间有连通图，且边都为交易较大的过程，就能初步判断为洗钱地址(对于交易所要进行判断，是中间结点还是始末结点)

4. 实现过程

1.读取vertex表的点数据,读取edge_table的边数据

Scala

```scala
//获取边数据
val edgeSource: DataFrame = spark.sql("select * from blockfilecoin.edg_table")

//获取点的数据
val vertexSource: DataFrame = spark.sql("select * from blockfilecoin.vertex")

//边和节点转化为rdd
val edgeRDD: RDD[ChainEdge] = edgeSource.as[ChainEdge].rdd

val vertexRDD: RDD[ChainVertex] = vertexSource.as[ChainVertex].rdd
```

2.根据有钱即有罪规则清洗点和边的数据

Scala

```scala
/*
 对于总共交易额少于100FIL以及每笔交易少一10FIL的边过滤
*/
//过滤不合要求的边
val edgeFilter: RDD[ChainEdge] = edgeRDD.filter(data => {
  val seed: Double = Math.pow(10, -18)
  val max_value: Double = data.max_value.getOrElse(0.0)
  val total_value: Double = data.total_value.getOrElse(0.0)

  //把总共转账少于100FIL和最大转账少于10FIL的边过滤，为非洗钱边
  if (max_value * seed >= 10 && total_value * seed >= 100) {
    true
  } else {
    false
  }
})
```

Scala

```scala
/*
对于交易总收入或者支出少于500FIL，每笔最大交易少于10FIL，余额为负数的结点进行过滤
 */
val vertexFilter: RDD[ChainVertex] = vertexRDD.filter(data => {
  val seed: Double = Math.pow(10, -18)
  var get_max_value: Double = data.get_max_value.getOrElse(0.0)
  var get_total_value: Double = data.get_total_value.getOrElse(0.0)
  var pay_max_value: Double = data.pay_max_value.getOrElse(0.0)
  var pay_total_value: Double = data.pay_total_value.getOrElse(0.0)
  val extra_money: Double = data.extra_money.getOrElse(0.0)

//假如节点的收入和转出的FIL都小于100FIL，最大交易额度小于1FIL的节点过滤，为非洗钱节点

//假如节点的收入和转出的FIL都小于500FIL，最大交易额度小于10FIL的节点过滤， 为非洗钱节点
if ((get_max_value * seed >= 10 && get_total_value * seed >= 500)
  || (pay_max_value * seed >= 10 && pay_total_value >= 500)) {
  true
} else {
  false
}
```

3.根据犯罪必有同犯的规则，对结点和边进行图连接

Scala

```scala
//建立ChainGraph图
val chainGrap: Graph[ChainVertex, ChainEdge] = Graph(vertex, edge)

//获取连通图
val verticesConnectedGraph: VertexRDD[VertexId] =
chainGrap.connectedComponents().vertices

//连接节点的信息
val verticesConnectedJoin: RDD[(VertexId, (VertexId, ChainVertex))] =
verticesConnectedGraph.join(vertex)
```

4.将数据写入hive中，key为graph_id

Scala

```
verticesConnectedJoinResult.toDS().createOrReplaceTempView("tmp_table")


spark.sql(
  """
    |insert into blockfilecoin.Vertex_ConnectedGraph
    |select *
    |from tmp_table
  """.stripMargin)
```

5.犯罪必有同犯，所以一个连通图中最少有两个结点，过滤只有一个结点的图

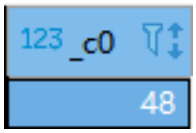
Scala

```
SELECT *
from (
     SELECT COUNT(*) as count, collect_set(filecoin_address)
     from blockfilecoin.vertex_connectedgraph
     group by graph_id
     HAVING count > 1
     ) tmp;
```

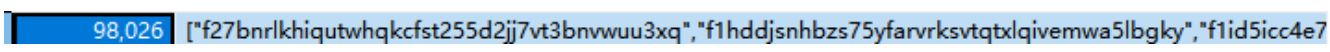
6.结果（第二列为相关异常地址的交易流程）

2	["f1utcyfhh3baqlnhr3yp4qvfuvbsg4bwrirwwwwry","f3vxrrbi7et4mnwzrcerv77q2cerpigh3olqtkwvbsnhtbwo7k
2	["f1tcnwmnzvy4wifg3eo5ss77x2di63k6v6ns35zfq","f15hskfb6irydhe436v2ds53b6rvlpdpa5siv6qyy"]
2	["f15qg4mao5r6p3glevb3agbyhkydihezlbqo53rti","f1t3o2dgcvkzcbnwuvgsn5nza3tr2xsgj3ho6v7sy"]
2	["f1zb6k2knu4dub72bksrcm6o335ef3mjp2nllxsyy","f1xmioo24j3moayunq3mzrklkfkkfokwntykrhloa"]
2	["f1ckk7v5bl6j52lt7yjtngdnlnzkiytb6tmqbetaq","f1gfj7tpxrutowksetd5k7un2ezkxibrcsv54eh7q"]
2	["f1ap4vf6msykzuv32clze7mzoyen6iiwmxjzpkxfi","f1ry65ti2kzgx4p5si2d6j5ulnzyzqdfeya5od2ra"]
2	["f16cuwl4ejwgtfcnj2iy4zij2vjzvlvwbvz4b54ry","f1zlmmhc6datrxik6hdx66mx5tq7gj25h2xwy5ora"]
2	["f17jepxswpkwbklhalep7trqq2crqwvmdzpvrjaua","f1gq5dwwxowreffa2nv4nax3ta7vooqr7fzz3rh3q"]
2	["f1cwy5i5wzai3sxgsjzhu6tu44wgr4qqsadkg5q5q","f1odspckvldhuqvzmlimxq3wnmqmkk7nsyq3yiwvi"]

从十亿行地址中切割出了48个连通图，其中47个连通图（交易图谱），图内节点数为2-15个，有一个图节点数有98026个，需要重新进行分析拆分。



123 _c0
48



98,026	["f27bnrlkhiqutwhqkcfst255d2jj7vt3bnvwuu3xq","f1hddjsnhbzs75yfarvrksvtqtxlqivemwa5lbgky","f1id5icc4e7

7.结论:图中节点交易量都是超500FIL相关的节点，其中有一个子图中有98026个地址的交易量都是超过500FIL,每笔交易量巨大，交易密集，需要重点分析。

8.改进：1.节点属性没有分类，如交易所节点，挖矿节点，需要了解。

2.其中一个连通图有98026个节点需要采取更为严格准确的方法进行分析处理

3.图中有些地址会有多个地址ID，造成地址总信息分裂，图断裂，造成巨大误差。